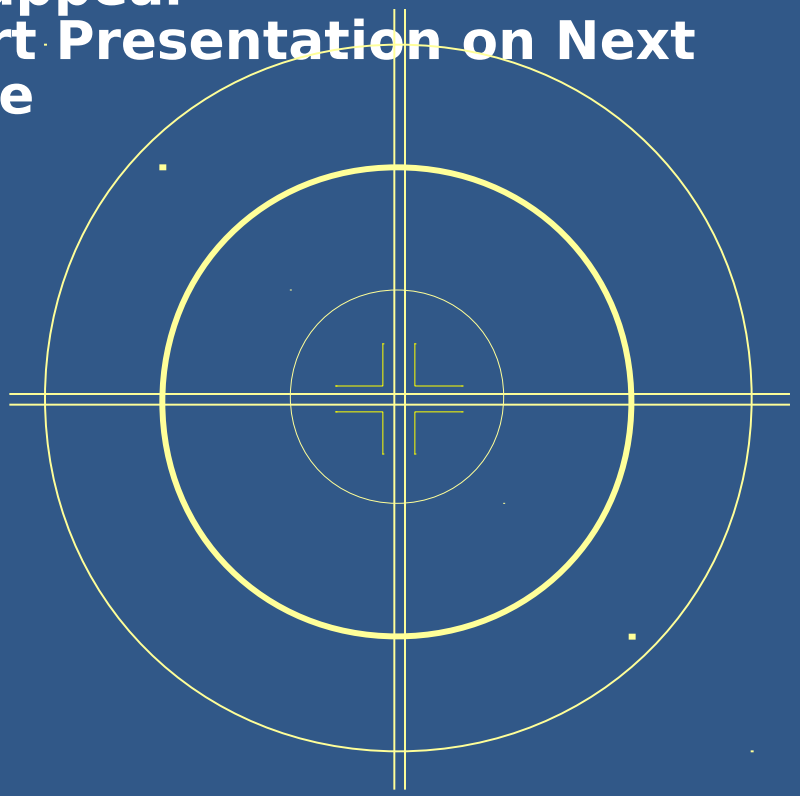
Set Up

1. **Change Computer Display Settings to “True Color”**
2. **Adjust Projector to This Slide**
3. **Above “Mask” there is a Blue Box – Adjust Brightness and Contrast to Make Box Disappear**
4. **Start Presentation on Next Slide**



FOCUS

MASK

COLOR

Mission Tool

Design Review

Nov. 14, 2001

6100 Chevy Chase Drive • Laurel, MD 20707 USA • Phone (301) 483-8900 • Fax (301) 483-8901

System Architecture

Esprit/TDF Relationship

- **Esprit Client GUI built upon TDF**
  - Many features for “free”
  - Code base used over many projects
    - Supports stability
    - Broader user base increases likelihood of bug discovery/resolution
- **Interfaces**
  - Allows interaction with classes without specific knowledge of individual class
- **Application base classes allow rapid development**
  - No re-inventing the wheel for basic application mechanics
- **Plugins**
  - No changes to code base
  - Plugin mechanism allows new menu items, track interfaces, view objects, etc. to be inserted dynamically

Interface Based System

- **Many components used in system are interface driven**
  - **Allows flexibility and extensibility**
  - **Examples:**
    - **Track -- a tagging interface, i.e., any class can be a track**
    - **ViewPointer -- pointers with different behaviors can be swapped in**
    - **ClientModules -- allow access to the track database from various sources**
    - **View Regions: Selectable, Drawable, RollOver**
  - **Listener Mechanism**
    - **Allows custom components to get notified when events occur**
      - **user input**
      - **state events**

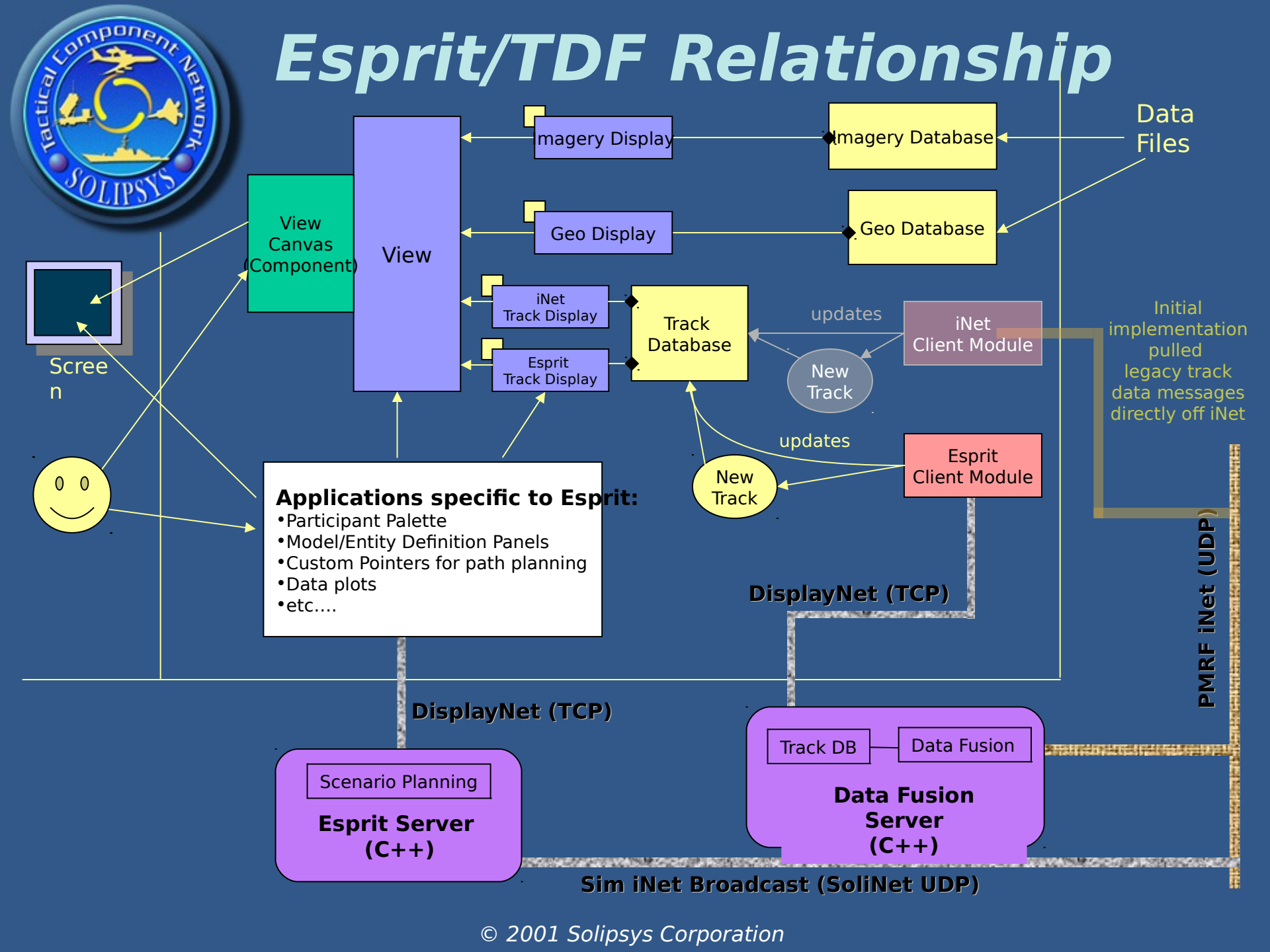
TDF Base Class Examples

- **Application**
  - Manages common functionality in sub-applications
    - DocumentApplication
    - FileApplication
- **BasicClientModule**
  - Manages common functionality in track interfaces
    - EspritClientModule
    - MSCTClientModule
- **AbstractModel**
  - Base class for data models
  - Allows multiple views independent of data

Esprit/TDF Plugins

- **How do Plugins actually work?**
  - **Any class can be a Plugin**
  - **Upon startup, DynamicLoader loads Plugin classes listed in plugin lists**
  - **Applications request all Plugins of a certain class from the DynamicLoader**
  - **Plugin applications can in turn load their own Plugins**
    - **For example, Preferences Application (a Plugin) dynamically loads PreferencePanels, which in turn may add Plugin components of their own**

Esprit/TDF Relationship



Current Capabilities

Current Capabilities

- Pre-Planning
- Planning
- Playback
  - Preview
  - Rehearsal
- Post Op

Scenario Planner: Preplanning

- Sensor model entry
- Vehicle model entry
- Entity definition
- Data set import
- Data set creation
- Overlay generation

Scenario Planning

- Vehicle entity assignments
  - Data sets
    - Imported
  - Waypoint definition
- Sensor entity assignment
  - Precision vs. Surveillance
  - Sensor coverage analysis
- Event timeline synchronization

Scenario Planning: Planning Aids

- Planning reports
- Radar coverage analysis
- Validity checks
- Time tics and synchronization lines
- Preview mode
- Multi-station playback

Event Rehearsal

- Event replay modes:
  - Preview
  - Multi-station playback
  - Extended execution (planned)

Event Monitor

- **Visual aids to:**
  - **Monitor real-time planned vs. actual position and path**
  - **Overlays used to delineate and monitor safety constraints**
  - **Decision aids provided to monitor:**
    - **Trajectory**
    - **Instantaneous Intercept Points (IIPs)**
    - **Aspect angle to shooter**
    - **Track analysis plots**

Use Case Overview for Baseline V3.1

- **Pre-Planning**
  - Define a Sensor Model
  - Define a Sensor Entity
  - Define a Vehicle Model
  - Define a TBM Model based on a “Duck” Dataset
- **Scenario Planning**
  - Create a Scenario
    - Add a TBM vehicle entity that uses the TBM model
    - Add a vehicle entity based on a non-TBM vehicle model and define its trajectory
    - Synchronize vehicle paths
    - Set T0
    - Set start times for non-synchronized paths
    - Place a sensor and assign sensor-tracking responsibility
    - Check signal-to-noise for a vehicle trajectory based on sensor assignments
    - Generate vehicle reports for use during the mission
  - Save a Scenario
  - Certify a Scenario (via a non-secure logon to server)

Use Case Overview for Baseline V3.1 (cont'd)

- **Scenario Execution**
  - **Preview a Scenario**
  - **Rehearse a Scenario using Simulated Radar Tracking**
  - **Monitor Real-time Events against Planned Scenario and Record for later Playback**

Use Case Analysis: GUI-Server Message Exchange

Pre-Planning Use Cases

- **Define a Sensor Model/ Sensor Entity/ Vehicle Model**
  - **Save as Working**
    - **No GUI-Server messages exchanged because working models and entities are saved on client**
  - **Save as Certified**
    - **<- sends GUIScenarioObject**
    - **-> receives SrvScenarioObjectResponse**
- **Define a TBM Model based on "Duck" Dataset**
    - **<- sends GUIConvertTextModel**
    - **-> receives SrvDataSetConversionCompleted**

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Create a Scenario**
  - **<- sends GUIScenarioControl, action = new scenario**
  - **-> receives SrvScenarioControlResponse, action = new scenario, status = success**
- **Add a TBM Vehicle Entity and Trajectory that uses the TBM model**
  - **<- sends GUIVehicle, action = new**
  - **-> receives SrvPlannedPathDump…**
    (for each waypoint on planned trajectory)
  - **-> receives SrvTimeTic**
- **Add Synch Points for TBM Trajectory**
  - **<- sends GUIVehicleAnnotation…**
    (for each synch point)

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Create a Scenario (Cont'd)**
  - **Add a Vehicle Entity based on a Non-TBM Vehicle Model and Define its Trajectory**
    - **<- sends GUIVehicleModel, action = new**
    - **<- sends GUIVehicle, action = new**
    - **<- sends GUIWaypointPosition, action = first**
    - **-> receives SrvWaypointPath**
    - **-> receives SrvWaypointPosition**
    - **For each point added via the GUI:**
    - **<- sends GUIWaypointPosition, action = successor**
    - **-> receives SrvWaypointPath…**
      (for each path segment)
    - **-> receives SrvWaypointPosition…**
      (for each waypoint)
    - **-> receives SrvTimeTic**

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Create a Scenario (Cont'd)**
  - **Delete Initial/ Following Waypoint**
    - **<- sends GUIWaypointPosition, action = delete**
    - **-> receives SrvWaypointPath…**
      (for each path segment)
    - **-> receives SrvWaypointPosition…**
      (for each remaining waypoint)
  - **Extend Path Forward/ Insert Succeeding Waypoint**
    - **<- sends GUIWaypointPosition, action = successor**
    - **-> receives SrvWaypointPath…**
      (for each path segment)
    - **-> receives SrvWaypointPosition…**
      (for each waypoint)

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Create a Scenario (Cont'd)**
  - **Extend Path Backward/ Insert Preceding Waypoint**
    - **<- sends GUIWaypointPosition, action = predecessor**
    - **-> receives SrvWaypointPath…**
      (for each path segment)
    - **-> receives SrvWaypointPosition…**
      (for each waypoint)
  - **Modify Existing Waypoint (by dragging or via InfoPanel)**
    - **<- sends GUIWaypointPosition, action = move**
    - **-> receives SrvWaypointPath…**
      (for each path segment)
    - **-> receives SrvWaypointPosition…**
      (for each waypoint)

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- Create a Scenario (Cont'd)
  - Synchronize Paths
    - <- sends GUIVehicleSynch
    - -> receives SrvPlannedPathDump… (for each waypoint on trajectory)
    - -> receives SrvVehicleAnnotation… (for each synch point)
    - -> receives SrvVehicleSynch, action = add
  - Set T0
    - <- sends GUITicReferencePoint
    - -> receives SrvTimeTic… (for each vehicle synchronized to that tic point)
  - Set Start Times of Non-Synchronized Paths
    - <- sends GUIVehicleModel, action = update
    - <- sends GUIVehicle, action = update
    - <- sends GUIVehicleStartTime, action = update
    - -> receives SrvWaypointPosition… (for each waypoint to be synchronized)

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Create a Scenario (Cont'd)**
  - **Place a Sensor and Assign Sensor-tracking Responsibility**
    - **<- sends GUISensorModel, action = new**
    - **<- sends GUISensor, action = new**
    - **<- sends GUISensorModeAssignment**
  - **Check Signal-to-Noise for Vehicle Trajectory based upon Sensor Assignments**
    - **<- sends GUIRequestSNvsTime**
    - **-> receives SrvSNvsTimeResponse**
  - **Request Vehicle Report**
    - **<- sends GuiRequestVehicleReport**
    - **-> receives SrvVehicleReportCompleted**

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Save Scenario**
  - **<- sends GUIScenarioObject**
  - **<- sends GUIScenarioControl, action = save working scenario**
  - **-> receives SrvScenarioObjectResponse**
  - **-> receives SrvScenarioControlResponse**
- **Close Scenario**
  - **<- sends GUIScenarioControl, action = new scenario**
  - **-> receives SrvScenarioControlResponse, action = new scenario, status = success**
  - **Server sends messages to remove scenario information from the display:**

    -> receives GuiVehicle, action = delete,
    GuiSensor, action = delete,
    etc.

Use Case Analysis: GUI-Server Message Exchange

Scenario Planning Use Cases

- **Certify Scenario**
  - **Logon to Server (non-secure)**
    - **<- sends GuiCertifiedPassword, action = Request**
    - **-> receives SrvCertifiedPasswordResponse**
    - **<- sends GuiScenarioObject**
    - **<- sends GuiScenarioControl, action = SaveCertifiedScenario**
    - **-> receives SrvScenarioObjectResponse**
    - **-> receives SrvScenarioControlResponse, action= SaveCertifiedScenario, status=Success**

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Open Scenario**
  - **<- sends GUIScenarioControl, action = load working scenario**
  - **-> receives SrvScenarioControlResponse, action = load working scenario, status = start**
  - **Server sends scenario, vehicle, sensor, etc. info, e.g. the following for a TBM trajectory:**
  - **-> receives GUIVehicleModel**
  - **-> receives GUIVehicle**
  - **-> recieves SrvPlannedPathDump… (for each point on the trajectory)**
  - **-> receives GUIVehicleAnnotation**
  - **-> receives GUIScenarioObject**
  - **-> receives SrvScenarioControlResponse, action = load working scenario, status = success**
  - **-> receives SrvScenarioObjectResponse**

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Preview Scenario**
  - Open Scenario
  - Select Playback Mode = Preview
    - [NONE]
  - Initiate a Scenario Timeline by Specifying a Scenario Time to Start
    - <- sends GUIScenarioControl, action = set scenario start
    - -> receives SrvScenarioControlResponse, action = set scenario start, status = success
  - Play Scenario
    - <- sends GUIScenarioControl, action = set clock rate
    - -> receives SrvScenarioControlResponse, action = set clock rate, status = success
    - <- sends GUIScenarioControl, action = start preview
    - -> receives SrvScenarioControlResponse, action = start preview, status = success
    - -> receives SrvScenarioSynch
    - -> receives SrvPlannedTrackState messages

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Preview Scenario (cont'd)**
  - **Hook Planned Position**
    - **[No GUI-Server messages – this is a TDF function]**
  - **Display Alt Graph**
    - **[No GUI-Server messages – this is a TDF function]**
  - **Stop Scenario**
    - **<- sends GUIScenarioControl, action = stop**
    - **-> receives SrvScenarioControlResponse, action = stop, status = success**

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Rehearse Scenario using Simulated Radar Tracking**
  - **Open Scenario**
  - **Activate Track Simulation using iNet Message 25**
    - **<- sends GUIScenarioControl, action = set sim type**
    - **-> receives SrvScenarioControlResponse, action = set sim type, status = success**
  - **Initiate a Scenario Timeline by Specifying a Scenario Time to Start**

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Rehearse Scenario using Simulated Radar Tracking (Cont'd)**
  - **Play Scenario**
    - **<- sends GUIScenarioControl, action = set clock rate**
    - **-> receives SrvScenarioControlResponse, action = set clock rate, status = success**
    - **<- sends GUIScenarioControl, action = start local execution**
    - **-> receives SrvScenarioControlResponse, action = start local execution, status = success**
    - **-> receives SrvScenarioSynch**
    - **Server sends iNet messages that are processed by Data Fusion and transformed into TrackUpdate messages**
      -> receives TrackUpdate messages
    - **-> receives SrvDisplayTrackState messages**
    - **-> receives SrvPlannedTrackState messages**
  - **Hook Planned Position**
  - **Display Alt Graph**
  - **Stop Scenario**

Use Case Analysis: GUI-Server Message Exchange

Scenario Execution Use Cases

- **Monitor Real-time Events Against Planned Scenario and Record for later Playback**
  - **Data Fusion processes iNet messages MT25 & MT 26 and transforms them into TrackUpdate msgs**
    -> receives TrackUpdate
  - **Open Scenario**
  - **Synchronize Scenario Timeline to Mission Countdown Clock**
    - **<- sends GUIScenarioControl, action = set scenario start**
    - **-> receives SrvScenarioControlResponse, action = set scenario start, status = success**
  - **Record using the Track Recorder**
    - **[No GUI-Server messages – this is a TDF function]**
  - **Play Scenario**
  - **Display Alt Graph**

Current Status

- **FY ’99 Acceptance Test in September 1999:**
  - Included Transition Plan
  - Noted deficiencies repaired for Dec ’99 delivery
- **FY ’00 funding for:**

  - Operational assessment during three events

  - Limited funds available for enhancements
- **Current FY ’01 funding for**
  - Addition of 14 new required features
  - Event support

Software Development Process

Dispelling Popular Myths

- **We are Merely Prototyping**
- **We are Rogue Hackers from Hell**
- **We are Lacking Process**

The more things change - the more things stay the same.

Instilling Actual Fact

Hell wouldn't have us!

- **Intimately Familiar with Rigorous Development Techniques**
- **Bring Many Years of Experience to Solving "Real World" Issues**
- **Dynamic Nature of Staff Allows for Rapid Turnaround of Viable and Robust Solutions - Often Confused with "Prototyping"**
- **We do not get Bogged Down by Process**

The High Level View

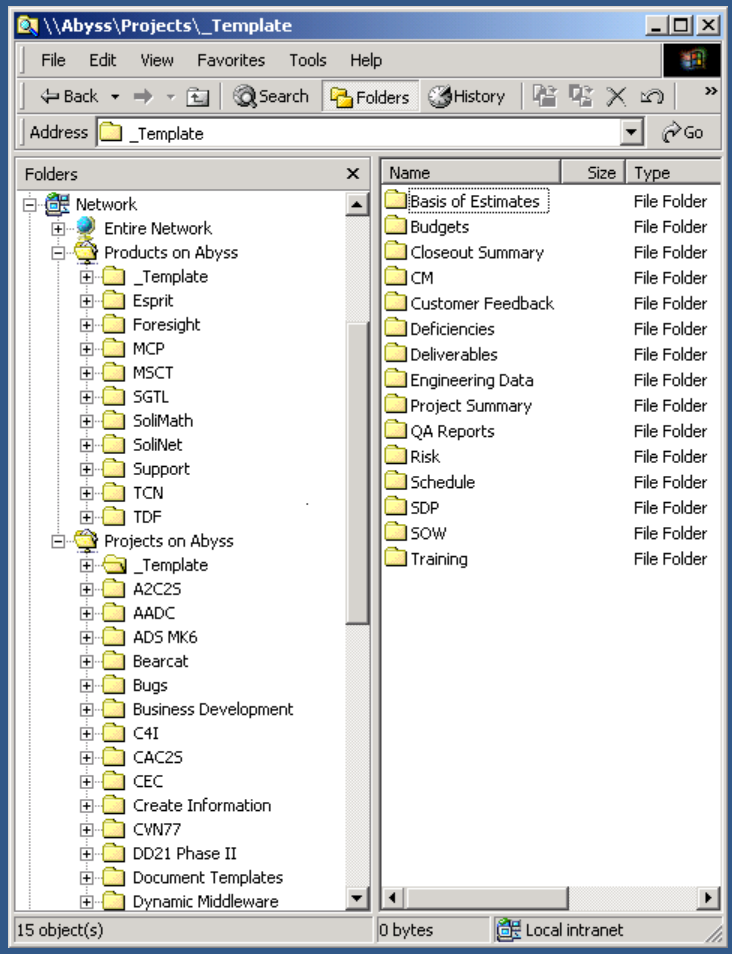
Standard approaches are taken to manage, track, and maintain the various products and projects.

- **Product and Project Documentation is Created and Stored on a Networked Server**
  - **Access to each folder is controlled by product or project manager**
  - **Each folder is accessible to the QA department for auditing purposes**
- ***Proceed* Tool Promotes a Uniform Structure Across Folders**
  - **Allows QA team to play a less intrusive role which makes everyone happy**
  - **Automated updates to push information to customer support area**
- **Development Suite is Comprised of Standard GNU Toolset and Custom Scripts**

View Into the Abyss

The product and project folders reside on abyss, files are backed up nightly, and off site storage is rotated weekly.

- **Primary Server is Named Abyss**
- **Provides Network Access to all Product and Project Folders**
- **System is Monitored and Maintained by System Admin Personnel**
- **Supports the Solipsys Proceed Tool**



Process Control with an Electronic Edge (Proceed)

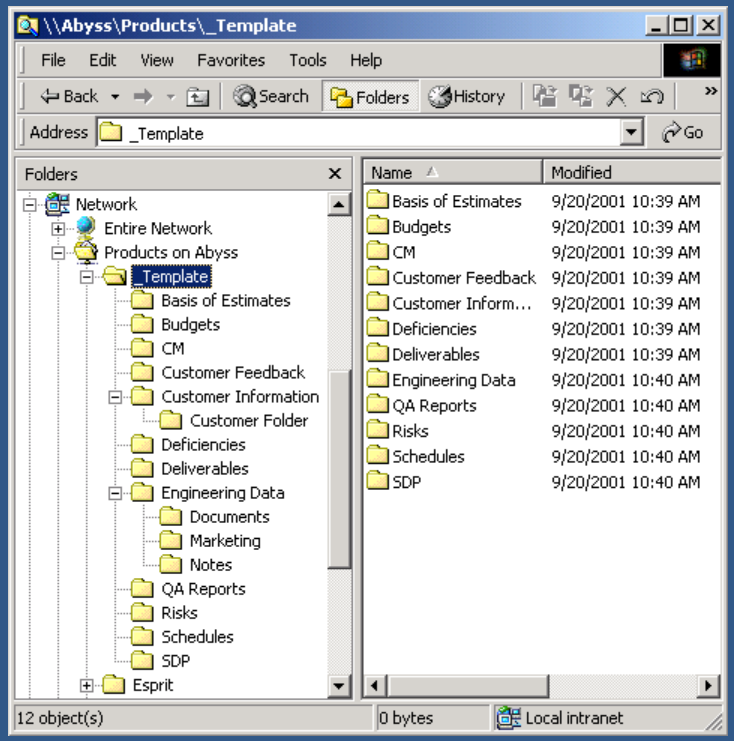
Process and QA are terms that are typically met with much resistance by those who view it as "the Man just beating me down."

- **Manages the Directory Hierarchies in the Products and Projects Areas**
  - Applies a directory structure template to all areas managed by Proceed
  - Performs consistency checks to ensure that the area is not being polluted
  - Does not inhibit, only guides
- **Pushes Desired Files from Configuration Area to Customer Support Web Site**
  - Managers may selectively allow any file in the hierarchy to propagate to the web site
  - Proceed will automatically update the files on the web site on a regular basis
- **QA Can Play a More Active Role Without Upsetting the "Volatiles"**

Proceed Hierarchy Template

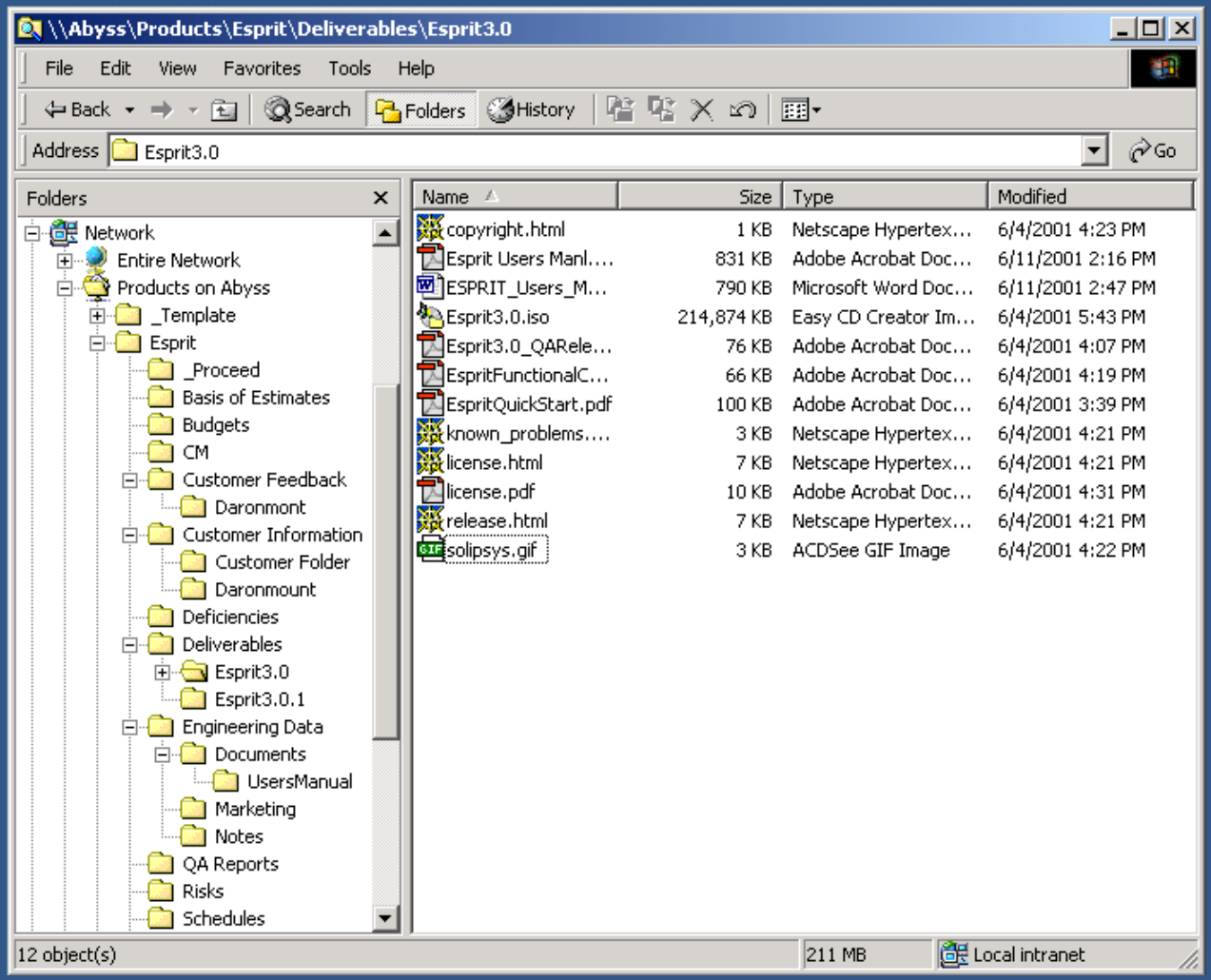
Proceed serves as an enabler to allow QA to lend a hand with maintaining a consistent and recognizable structure in a less invasive manner.

- **The QA Team Maintains the Hierarchy Template**
- **Proceed Ensures Hierarchical Consistency Across All Products and Projects Folders**



Sample Product Folder - Esprit

The Esprit product folder contains deliverable ISO images, licensing information, QA release notes, bug lists, etc.



Software Development Environment

The GNU toolset, automation of process, and reusable software components are the foundation of the software developmen t cycle.

- **Standard GNU Toolset is Employed**
  - Available for most platforms
  - Promotes portability
  - Widely supported by the Net community
  - Cost effective
- **Custom In-House Techniques**
  - Scold
  - Rebuild
  - Metagen and Cajun
- **Solid Foundations Through Software Reuse**
  - SoliNet
  - SoliMath
  - Granite
  - TDF

GNU Tool: Concurrent Versions System (cvs)

CVS is based on older Revision Control System (rcs); both are commonly used GNU packages.

- **Network Based Revision Control System**
- **Enables Multiple Developers to Work on Parts of the System**
  - **Reduces amount of “dead time” caused by a lock based system**
  - **May introduce conflicts if the exact same area is modified, but cvs supports conflict reporting and resolution**
- **Local Environment is Replicated on the Server at PMRF**
  - **Developers enjoy a familiar environment when in the field**
  - **Necessary for distributed testing and development**

GNU Tool: cvs (cont)

Concurrency, consistency, and conformity are our cornerstones to building and maintaining robust systems.

- **Supports Branching for Use When Configurations are Taken into the Field**
- **Baseline Configuration Project (cvsproject) is Used as the Genesis for All New Projects**
  - **Ensures a consistent look and feel across all projects in the system**
  - **Allows new projects to start with a robust and well tested configuration**
  - **The cvsproject is maintained and kept current**
- **Available for Solaris (Sparc/x86), various flavors of Linux, and Cygwin32**

GNU Tool: cvs (cont)

- **Maintains Status Logs and Modification Commentary**
- **May be Configured to use Custom Scripts for Automated Generation of Change Reports**

The sample change log is from the original mods to the SoliNet make and auto configuration files.

```
=================================================
Root: /thor/CVS                  User: john
08/18/98
Dir : solinet
15:18:41
File:  Makedefs.in Makefile.in Makerules.in
acsite.m4 archive
      configure configure.in  Mods:
  **   Massive changes went in to support a software
  **   installation. Most of the modifications were in
support
  **   of placing the SoliNet and SoliMessage
include files down
  **   one level rather then having everything in
/include. The
  **   changes to the make system were made in
support of adding
  **   SoliNet to the scold automated source code
checking
  **   scheme. All of the modifications made to the
make system
  **   were migrated to the cvsproject template
directory which
  **   is maintained under CVS control.
=================================================
Root: /thor/CVS                  User: john
08/18/98
Dir : solinet
15:32:06
File:  configure configure.in  Mods:
  **   Restructured the src/lib directories by
renaming solinet
  **   to SoliNet, solimsg to SoliMessage, and
metagen to
  **   MetaGen.
=================================================
```

GNU Tool: autoconf

Using autoconf and SoliNet completely divorces the software from the system.

- **Not all Unixes are Created Equal**
  - Ferreting out subtle differences between systems can be a configuration nightmare
  - Developing and maintaining portable scripts is a monumental task
- **Autoconf Allows configure.in Files to Specify High Level Tests**
  - The configure.in file is parsed and a configure script is generated using Bourne shell syntax
  - The resultant script can be executed on any Unix platform
  - The test results are applied to the make system
- **SoliNet Utilizes autoconf to Completely Isolate System Dependencies from the Application**

GNU Tool: make

Make is repsonsible for the compiler and linker interactions necessary to create executable code from the system's source code.

- **Make is Actually a Unix Utility Which GNU has Extended**
  - **Maintains dependencies between files**
  - **Understands how to rebuild pieces of the system based on an extensible rule set**
  - **Rules exist to utilize Solipsys meta language extensions**
- **Recursively Drills Through a Project's Directory Hierarchy Building the System Based on the Rules**
- **Employed by CVS Repositories for Rebuilding the World**

GNU Tool: gcc

Gcc is a fully ANSI compliant C/C++ compiler that is available on a wide range of architectures.

- ANSI Compliant C++ Compiler
- Used in Nit Pick Mode
  - Full type checking is performed
  - All non-standard constructs are flagged as warnings
  - All unsafe operations are flagged as warnings
  - All warnings are treated as errors and may not be ignored
- Make and Configuration System Controls the use of gcc
  - Users cannot disable the warning level imposed by the system
  - Users must fix all warnings and errors before a functional system may be completely built

In-House Tool: Scold

Scold acts as a source code and configuration watchdog by performing auto-builds on a regular basis.

- **The Source Code OverLordD (scold) is a Configuration Area as Well as a Set of Scripts**
- **Used as the Central Repository for Latest Builds**
  - **Obviates the need for each developer to build local copies of the most commonly used toolkits**
  - **Maintained by the configuration manager**
- **Also Refers to the Automated Scold Process**
  - **Periodic checkout and builds of the systems**
  - **Performs analysis of the logs from the build process**
  - **Flags problems and notifies last person that performed a modification to the problem unit**

In-House Tool: Rebuild

The rebuild script allows the build manager to easily build the configured software on any machine.

- **Perl Script that Understands the Package Build Order**
- **Provides a Simple One Line Command to Build Any or All of the Systems**
  - **Can rebuild the world from scratch**
  - **Can update and build only modified systems**
  - **Allows the build manager to selectively create specific versions of the systems**
- **Creates a Complete Log of the Build Process as Well as System Specific Breakdowns of What Took Place**

In-House Tool: Metagen and Cajun

Use of the meta languages allows the machines to do the dirty work!

- **Messages and Classes are Described in a Simple High Level Language**
- **Description Files Pass Through Meta-Language Compiler to Produce Complete C++ and Java Source Code**
- **Source Code is then Processed by the C++ and Java Compilers to Create Fully Functional Objects**
  - **Eliminates common mistakes in defining and maintaining message structures**
  - **Insures consistent message interfaces between all C++ and Java components within the system**

Software Reuse: SoliNet

SoliNet enforces strong typing and encapsulation of data, thus promoting solid object oriented designs.

- **An ANSI C++ Compliant Class Library Initially Created to Provide a Wrapper for Commonly Used Network Interfaces**
- **Took on the Role of Being the Operating System Layer Used by Emerging Solipsys Technologies**
  - **SoliMath**
  - **Granite**
  - **DataFusion**
  - **Esprit**
  - **TCN**
- **Evolved Into a Lightweight, Portable Operating System Layer with Extensive Support for Distributed Heterogeneous Communications**

Software Reuse: SoliNet *(cont)*

SoliNet is at the heart of the software being deployed at PMRF.

- **Foundation for the Esprit Scenario Planning Server**
- **Provides Logical Building Blocks and Network Infrastructure for the Range Upgrade Effort**
  - **Bridges the PMRF iNet to the Solipsys kNet through the FTCS**
  - **Foundation for the Source Integration Servers**

Software Reuse: SoliMath

Common math classes like filters, state estimation, covariances, bias estimation, bias measurment, and more reside in SoliMath.

- **Large Collection of Math Specific Classes**
  - Well over 100 classes currently reside in the toolkit
  - Functionality of classes backed by MatLab models developed and tested by analysis staff
  - Utilizes SoliNet as foundation therefore instances of SoliMath objects can be passed in messages
- **Utilized Heavily by Other Solipsys Project and Products**
  - Granite
  - DataFusion
  - Esprit
  - TCN
  - JLENS

Software Reuse: Granite

- **Graham Should Do This Slide??**

XXXXXXX

Software Reuse: Tactical Display Framework (TDF)

Use of Java technology and good OOP design enables the TDF to swiftly take on any new display role thrown at it.

- **Foundation for all Solipsys Graphical User Interfaces (GUI)**
- **Easily Tailored by use of Java Plug-In Technology**
- **Basis for Several Different Display Products at PMRF**
  - **Mk74 SIS**
  - **CoSip SIS**
  - **Surveillance SIS**
  - **Esprit**
- **Presents a Common Look and Feel Across the Various Operator Displays**

Meta-Compiler System

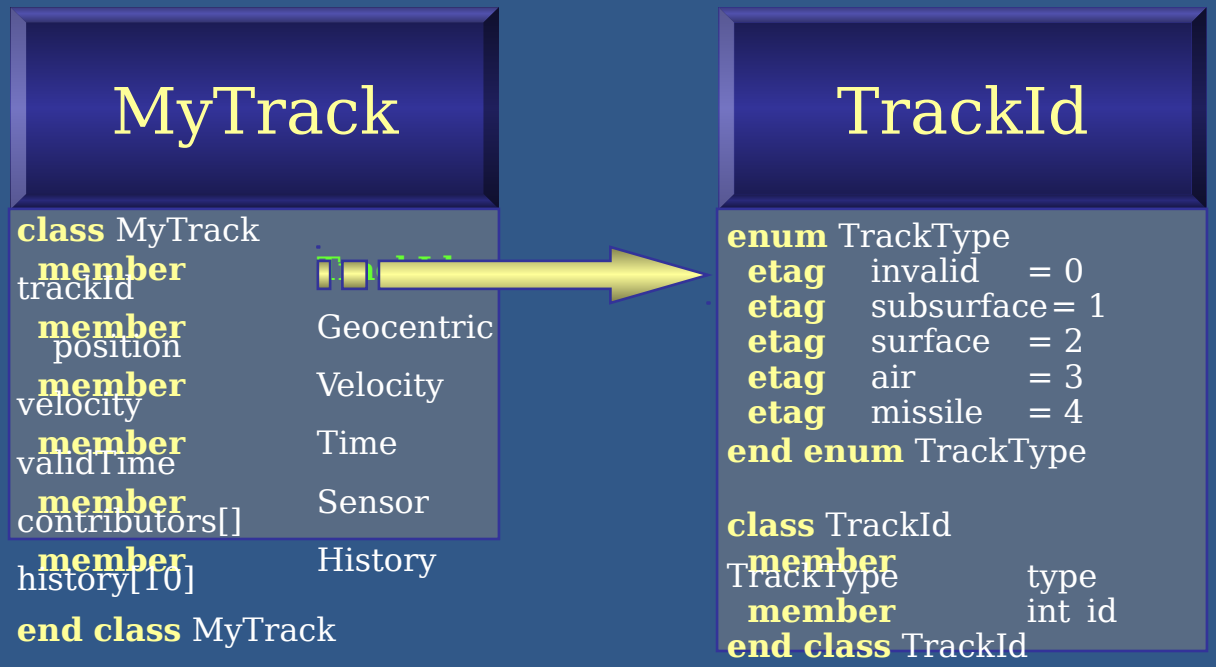
6100 Chevy Chase Drive • Laurel, MD 20707 USA • Phone (301) 483-8900 • Fax (301) 483-8901

Meta-Compiler in Action

Building a class with the meta language is fast, easy, and correct.

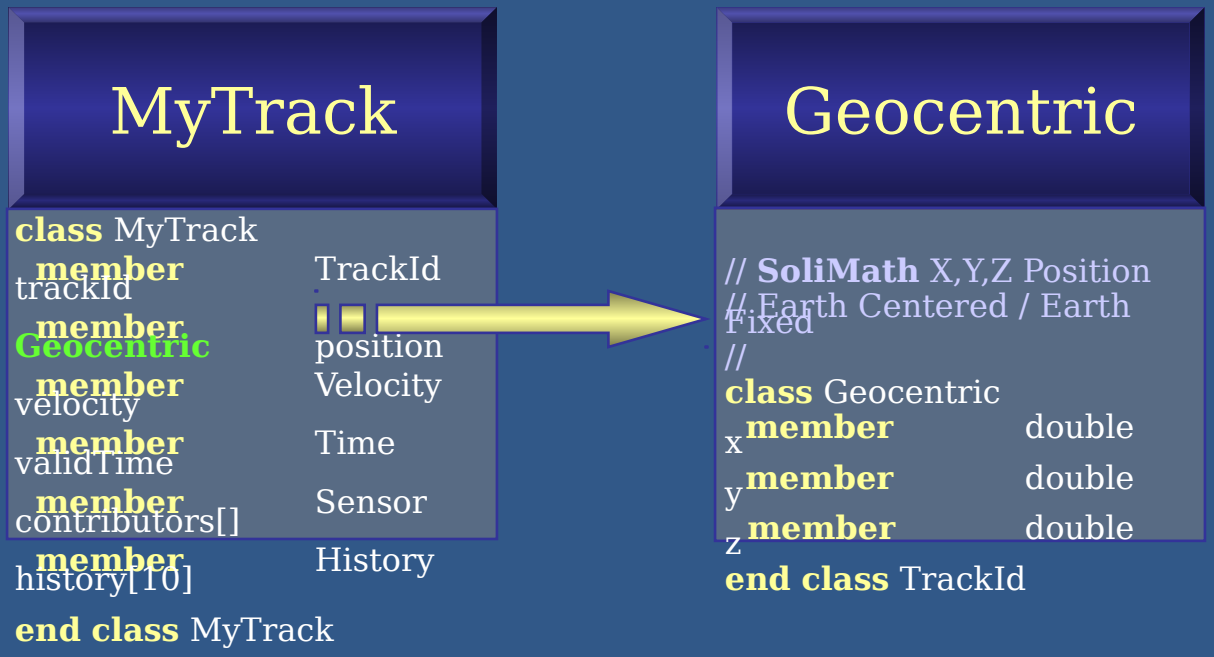
Defining Classes Within Classes

Classes may be nested as shown here with TrackId.



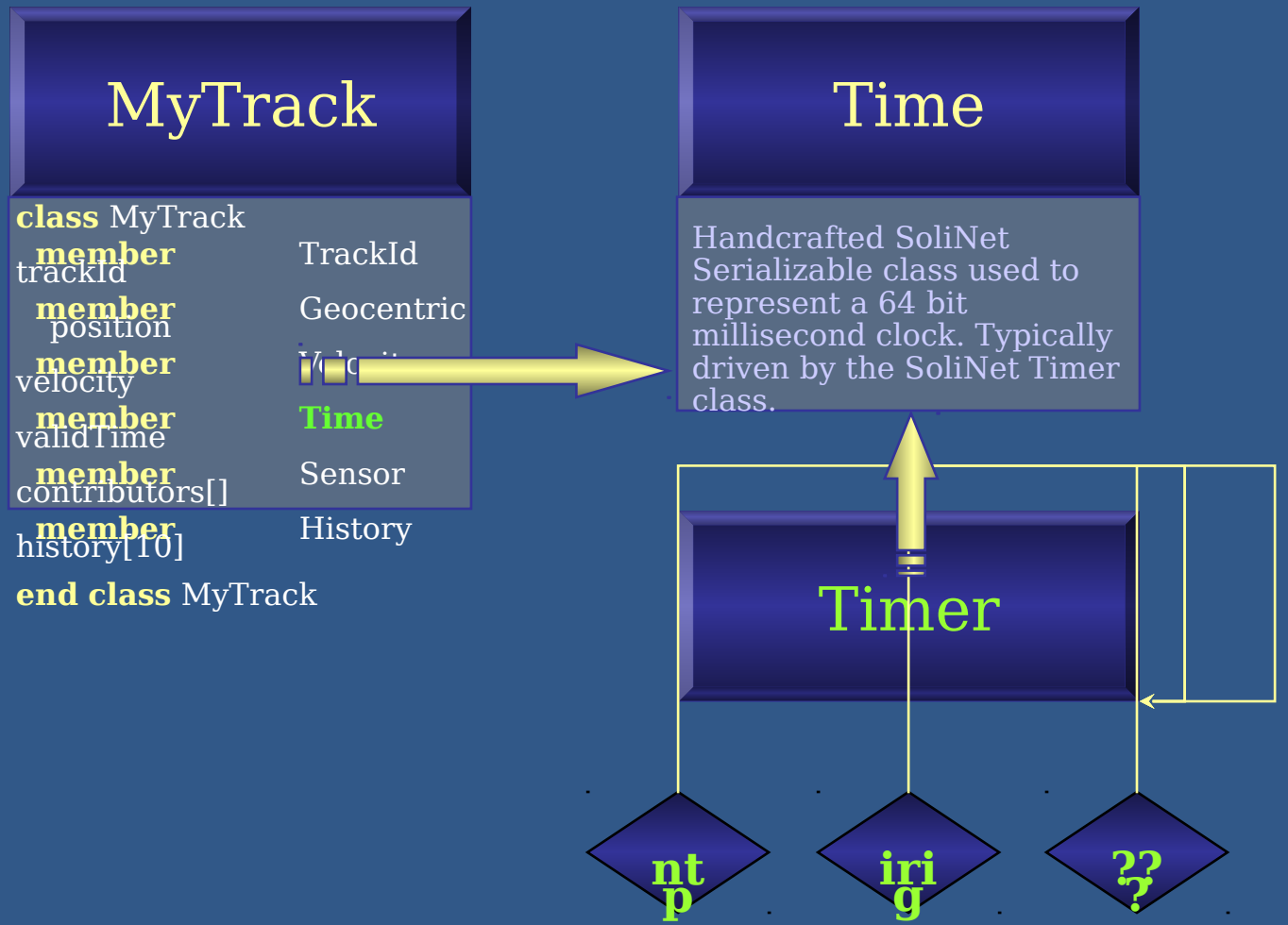
Using Pre-existing Classes

Classes from existing libraries may also be used to avoid reinventing the wheel.



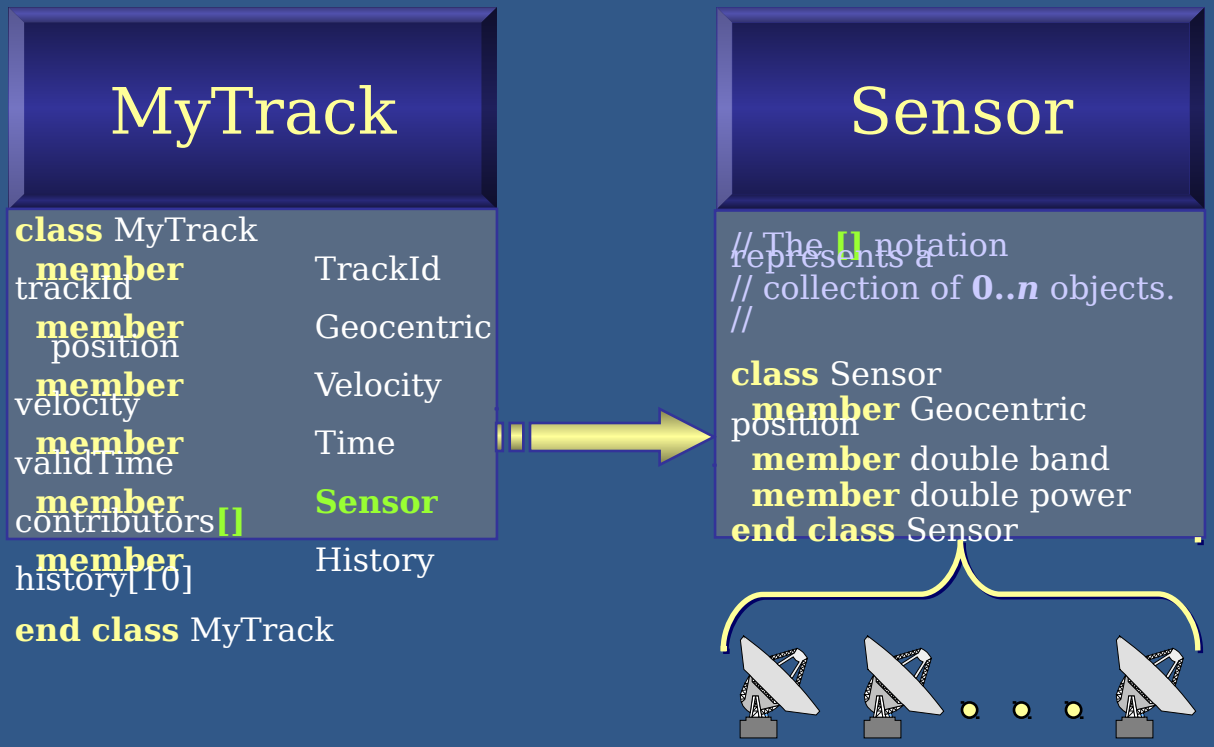
Utilizing SoliNet Time

The SoliNet Time class can be used to provide a simple and consistent notion of system, or scenario time.



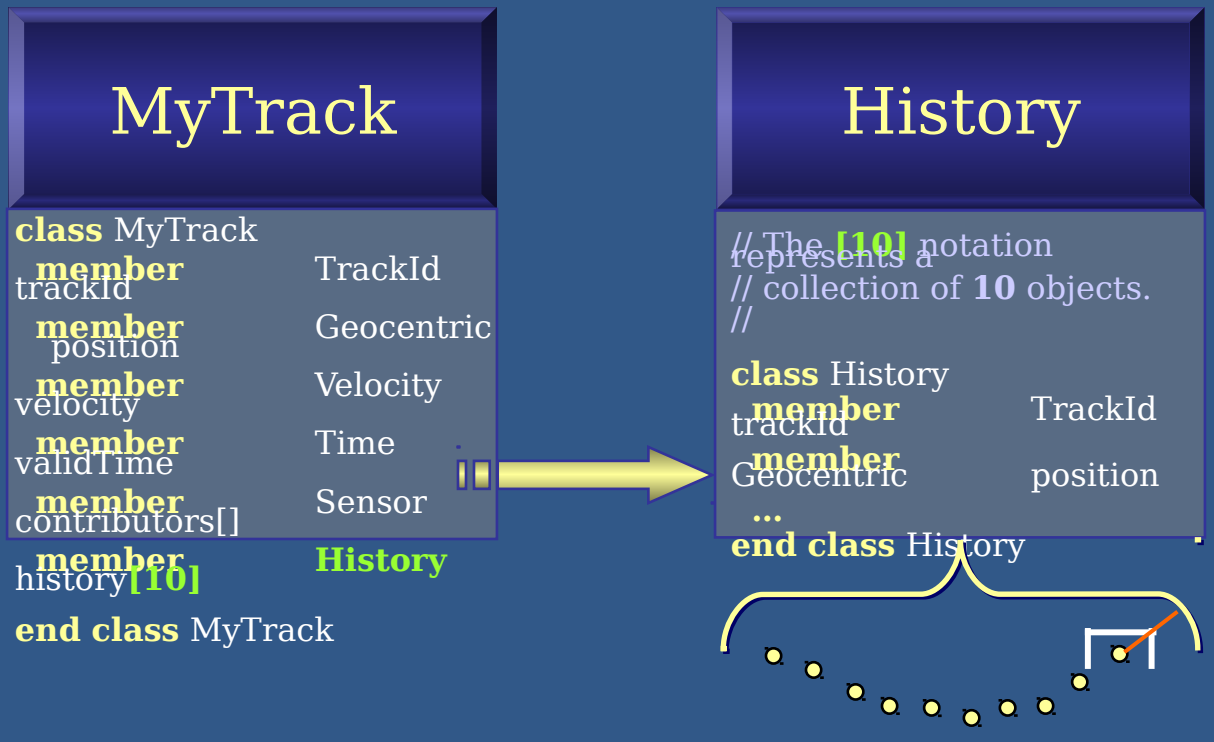
Collections: The List

Object collections, in this case an unbounded list, are supported by the meta language.



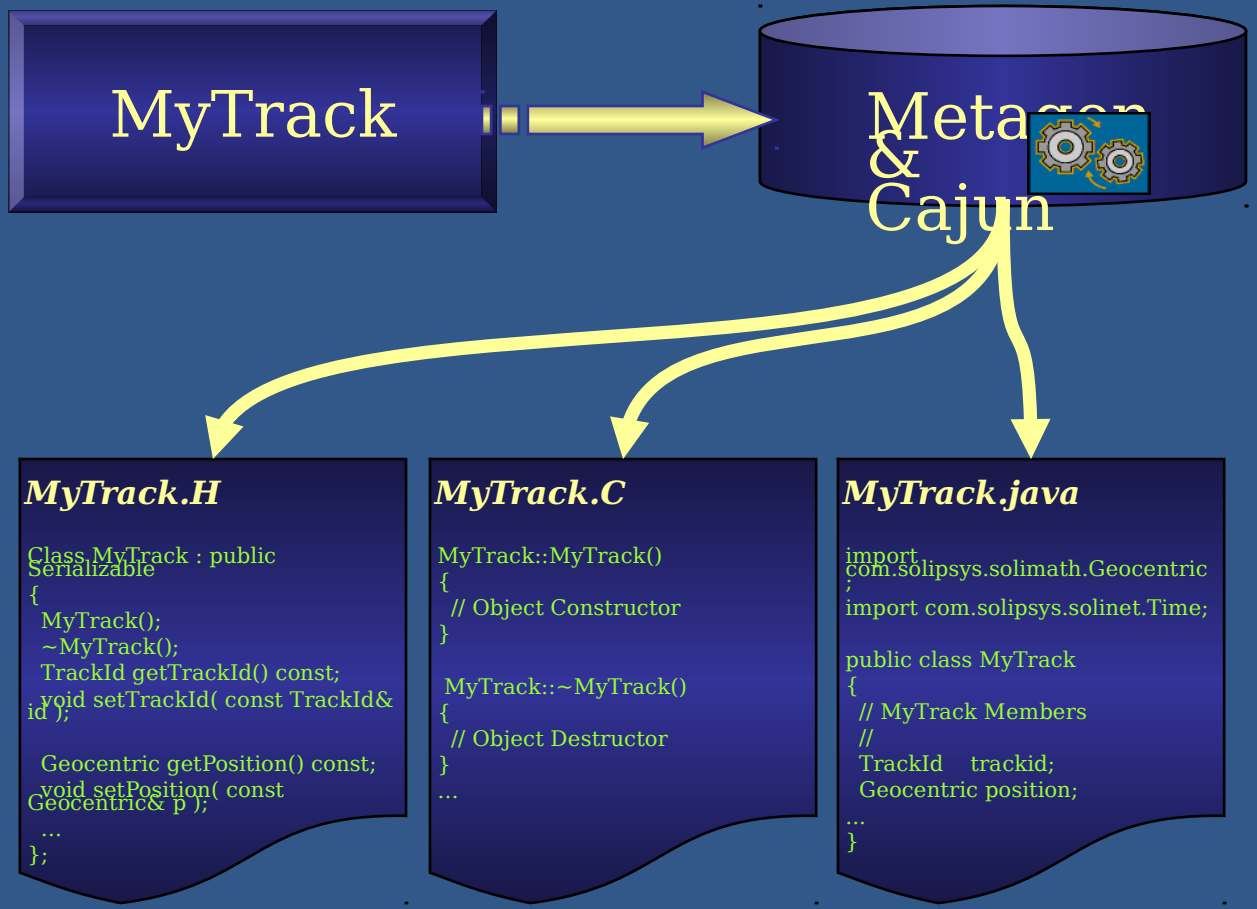
Collections: The Vector

Collections may also have hard boundaries placed upon them.



Code Generation Phase

Feeding the meta source into the compiler will produce all of the C++ and Java code needed to freely exchange the objects.



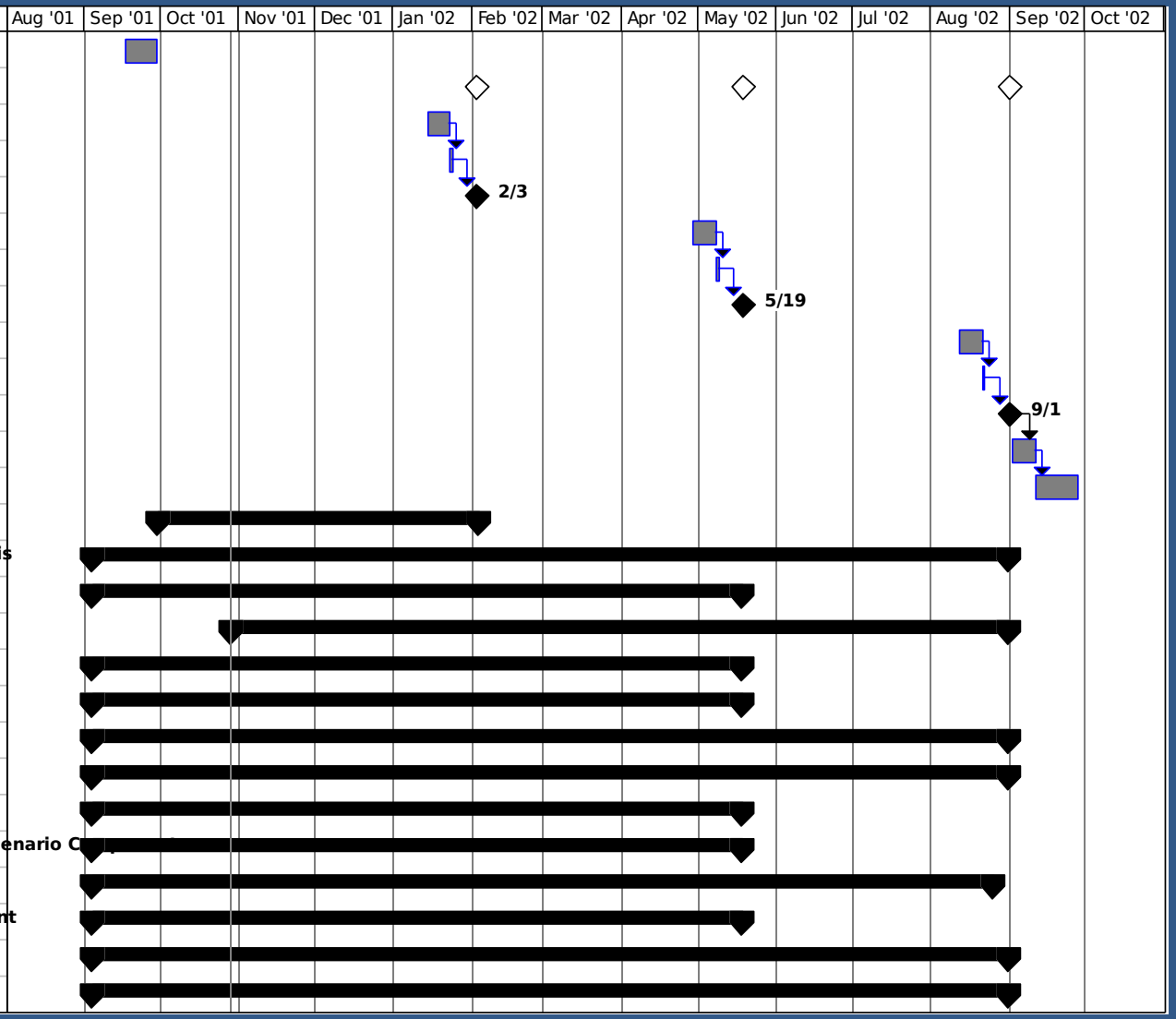
- ***Bullets***

Some text…

Software Development Plan

Software Development Plan

ID	Task Name
1	Baseline Demonstration
2	**Software Deliveries**
3	Baseline 1 QA Tests
4	Baseline Review
5	Software Deliveries 1
6	Baseline 2 QA Tests
7	Baseline Review
8	Software Deliveries 2
9	Baseline 3 QA Tests
10	Baseline Review
11	Software Deliveries 3
12	Acceptance Test
13	Acceptance Test Documentation
14	**1. CD Installation**
18	**2. X/Y Plots with Configurable Axis**
26	**3. Process iNet ASCII Data**
32	**4. PFPS Flight Paths**
39	**5. Participant Paths**
45	**6. Classification Banner**
51	**7. Decision Aid Matrix**
58	**8. Compute Real Time Events**
65	**9. Add RF Sensors to Participants**
71	**10. Translate, Rotate, and Flip Scenario C**
77	**11. Radar Coverage Analysis**
84	**9. Plan Configuration Management**
90	**13. Reports**
97	**14. Timeline Enhancements**



Software Development Plan

- **Show MS Project Plan**
  - **Build Schedule**
  - **QA**
  - **Delivery Schedule**
  - **Documentation**
  - **Acceptance Test**

Installation CD

Installation CD

- **Currently using InstallShield, Java edition**
  - **One installer for Win32, Solaris, etc.**
  - **Java based, requires a JVM pre-installed**
  - **Some minor issues, may be resolved with future InstallShield releases?**
- **Other options**
  - **Self-extracting zip file**
    - **Simplest option**
    - **Less flashy**
    - **Won't do system dependent tasks**
  - **Multiple installers for multiple systems**
    - **InstallShield/Wise for Win32**
    - **Solaris packages**
    - **Other platforms, e.g., Linux rpm**
    - **If automated, is this really more effort?**

Real Time Analysis Plots

Real-Time Data Presentation

- **Analysis Plots**

Plots

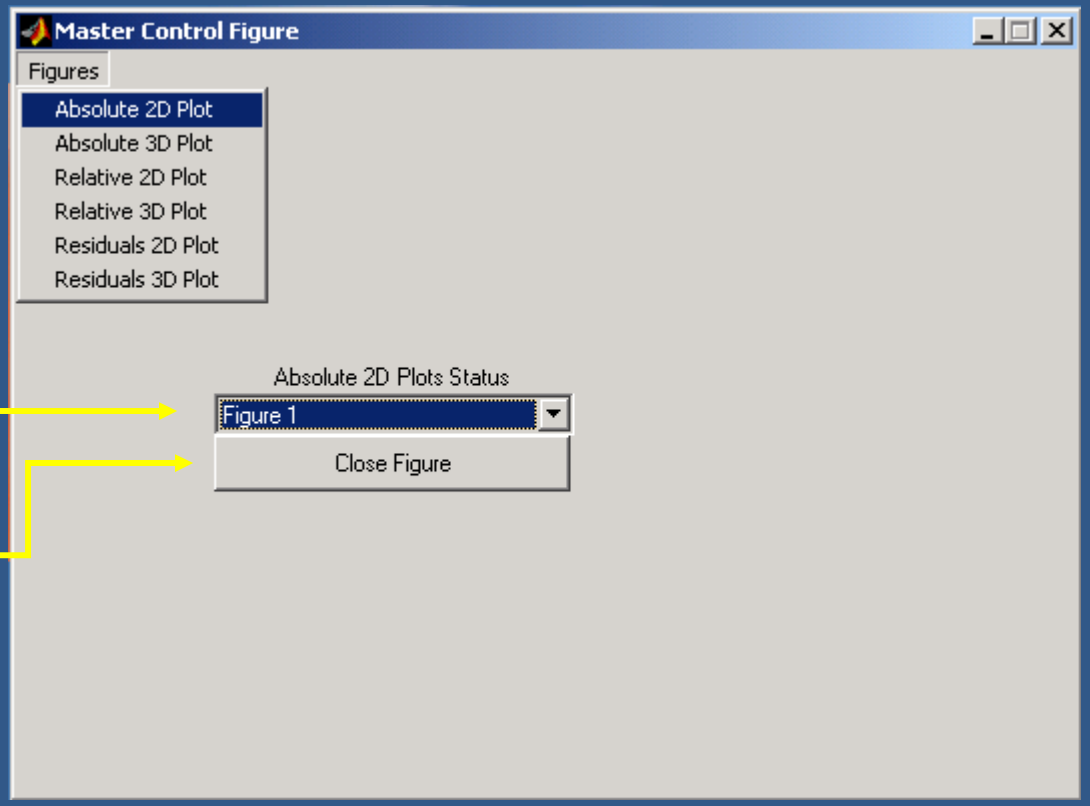
- iNet Kinematic Contents
  - Position, Velocity, Acceleration, Valid Time
  - Track State Covariance
  - Attitude
- iNet Attribute Contents
  - Track Quality
  - VID/SID
  - Security Level
- Derived Quantities
  - Down range, Cross range, Altitude
  - Aspect Angle
  - Relative spherical position
  - Closest Point of Approach (CPA)
  - Time-to-go
  - Residuals

Plots

Add New Figures

List of Current Figures

Close Selected Figure

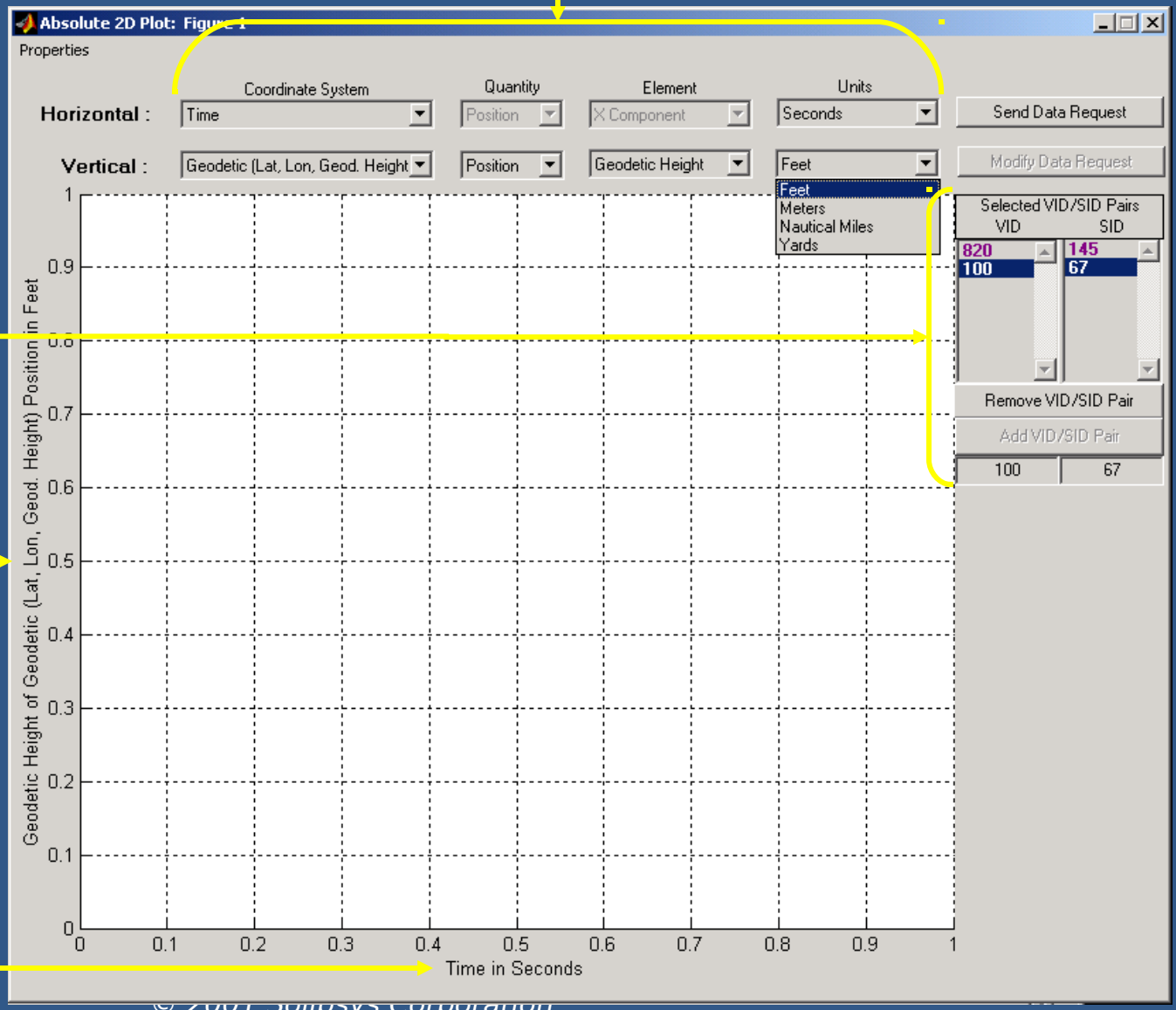


Plots

Adaptive Pop Menus

VID/SID Entry Panel

Adaptive Axis Labels



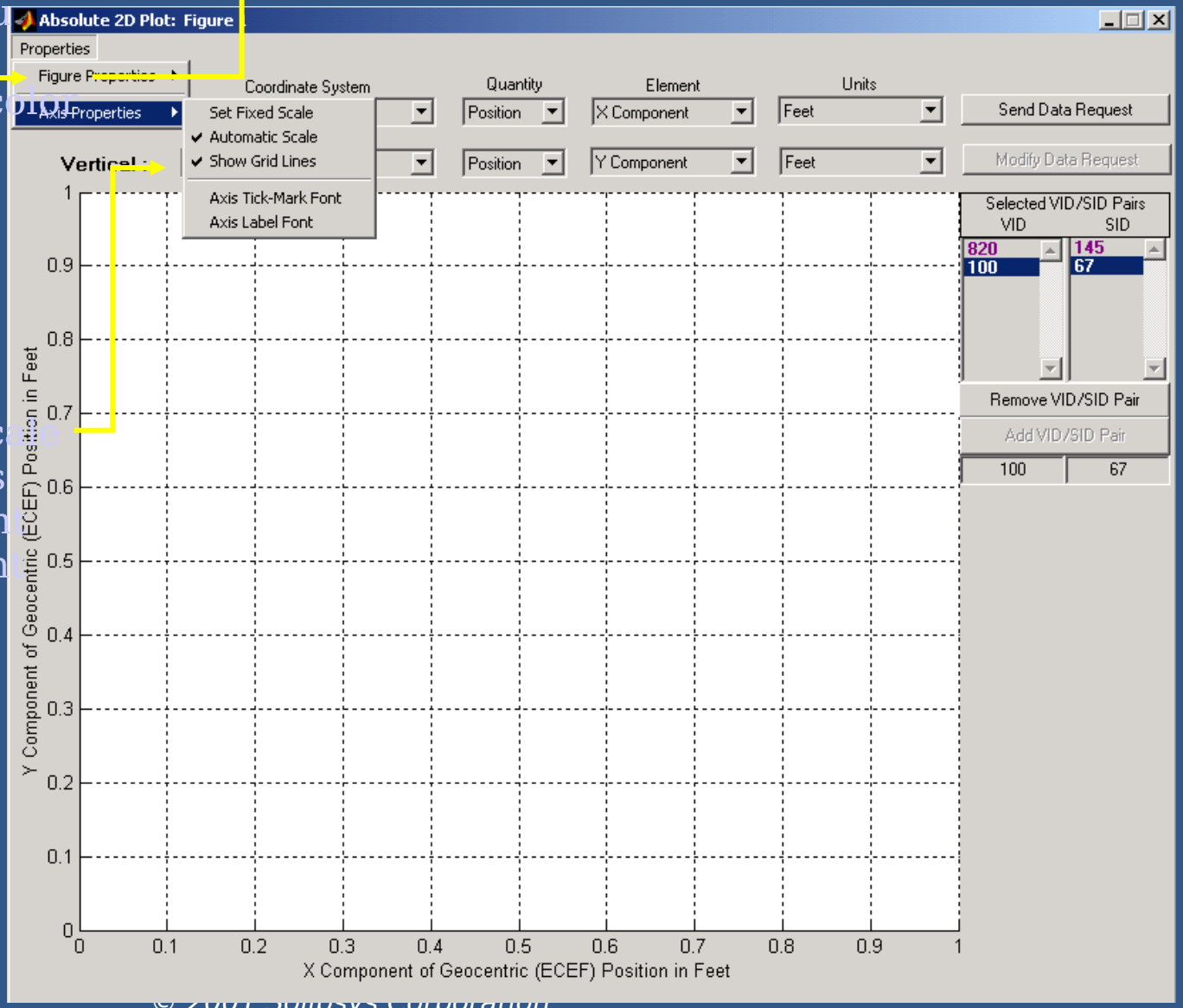
Plots

Figure Properties Menu

- Modify background color
- Save figure settings

Axis Properties Menu

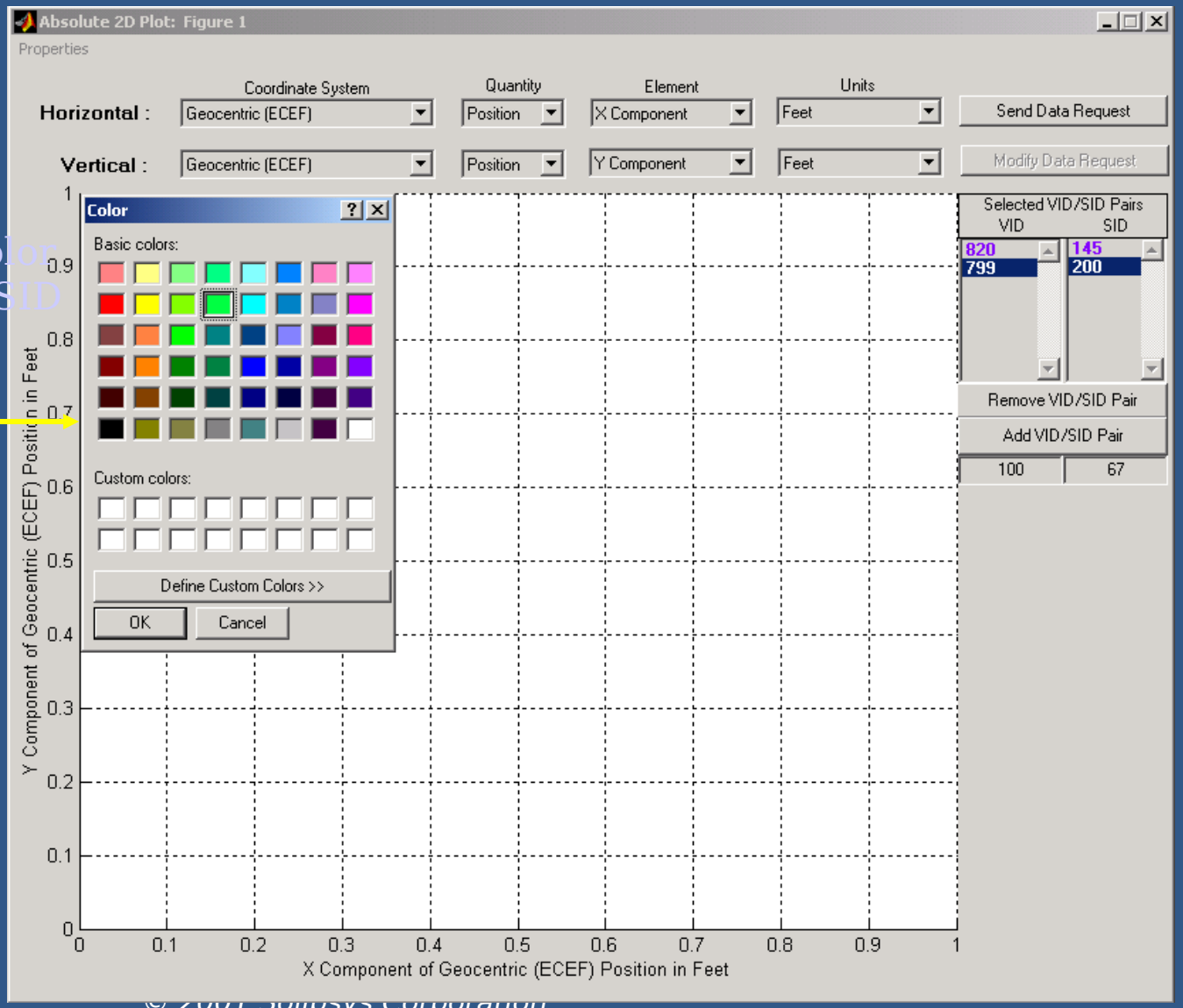
- Set fixed axis scales
- Turn on automatic scale
- Toggle axis grid lines
- Change tick mark font
- Change axis label font



Plots

Custom Color Panel

- Modify background color
- Assign color to a VID/SID



Real Time Analysis Plots
Build Schedule

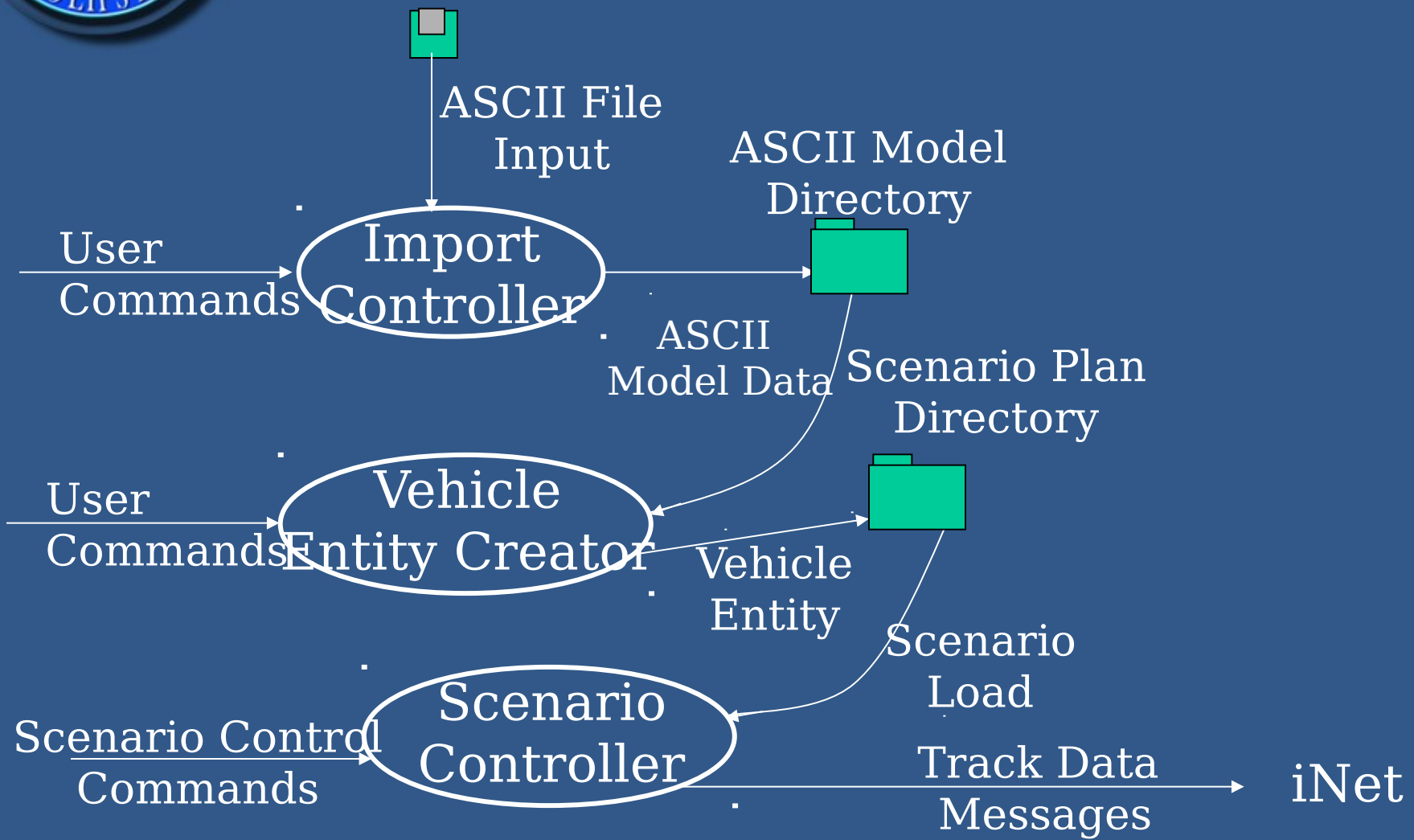
- **Build 1 – 2/3/02**
  - X/Y plots with fixed scales, units, and increments for all explicit and implicit track data.
- **Build 2 – 5/19/02**
  - Refinements
  - X/Y plots with fixed scales, annotations, and increments for additional plots that are reference point dependent.
- **Build 3 – 9/1/02**
  - Refinements

Process ASCII Formatted Trajectory Data

Process ASCII Formatted Trajectory Data
Description

- **Mission Tool shall import ASCII formatted trajectory data**
  - **Time, position, rates, optional orientation data**
- **ASCII model imported into any scenario as a vehicle entity**
- **Scenario execution sends requisite planned position and Track Data messages.**

Process ASCII Formatted Trajectory Data



Process ASCII Formatted Trajectory Data

Build Schedule

- **Build 1 – 2/3/02 Full Capability**
- **Build 2 – 5/19/02 Refinements**

Personal Flight Planning System (PFPS) File Interchange

PFPS File Interchange

- Recently received requisite Common Route Definition (CRD) Format - from TAMPS IDD
- Is this a good time to review the genesis and merit of this requirement?
- Recommend additional study and white paper presentation for this phase of the implementation

Vehicle Waypoint Trajectories

Special Cases

Participant Locations as Fixed Points

- Many participants are positioned at a specific location throughout a scenario
  - Aux Sensors
    - MATSS
  - Interceptor launch vessel
    - USS Lake Erie
- Others follow a simple racetrack to remain near a specific location
  - G1 aircraft
  - P3 aircraft
- New feature will allow these participants to be added easily

Stationary Participants

- User selects a vehicle, assigns a VID, and places into scenario
- If vehicle has only one waypoint, and minimum speed is zero, user can right click on waypoint and designate as stationary
- By default, vehicle will exist throughout scenario
  - If desired, start/stop times may be entered

Racetrack Participants

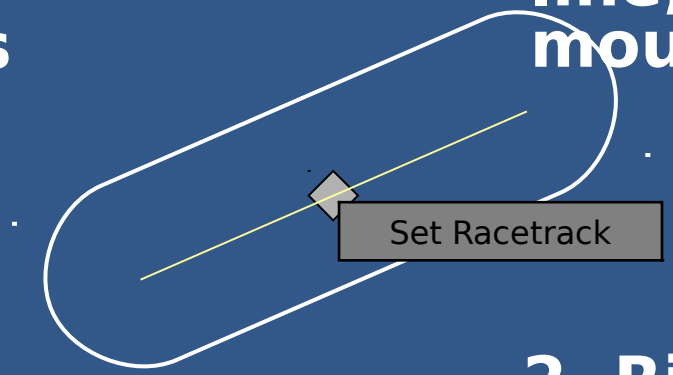
- User selects a vehicle, assigns a VID, and places into scenario
- If vehicle has only one waypoint, user can right click on waypoint and designate as racetrack
  - Axis line appears through waypoint to cursor
  - Mouse click sets axis line
  - Waypoint turn radius determines minimum axis length
- By default, vehicle will exist throughout scenario
  - If desired, start/stop times may be entered

Racetrack Entry

1. User enters Waypoint

2. Right-click popup menu item

3. Drag out axis line, and click with mouse



4. Racetrack is drawn

Build Schedule

- **Build 1 – 2/3/02**
  - **Full Capability**
- **Build 2 – 5/19/02**
  - **Refinements**

Classification Banner

Classification Banner Design

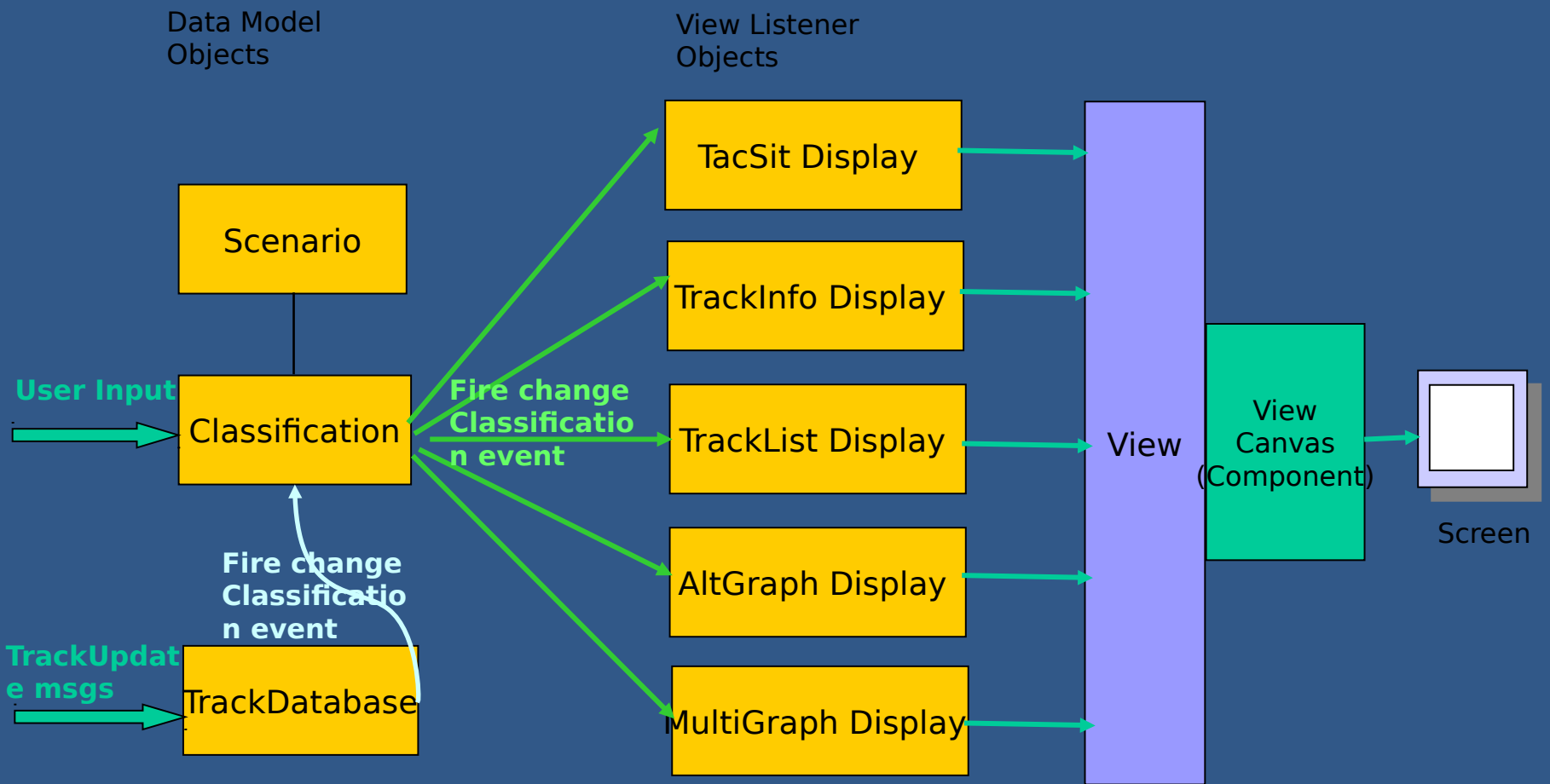
- **Description**
  - User shall be able to attach a classification level to any scenario
  - Classification level shall be displayed prominently on all displays and generated reports
- **Design Considerations/ Issues**
  - When monitoring real-time events, need to alert user if live track message received with security level higher than what is currently displayed in classification banner
  - Any frames that are printed via the GUI need to include classification level
  - Does the password sent to change classification level need to be encrypted when stored on Server and sent between GUI and Server?

Classification Banner Design

- **Data Flow**
  - GUI and Server will use GUIScenarioControl message to exchange classification info
    - Add new action SaveScenarioClassification
    - Add classificationLevel field
    - Add changeClassificationLevelPassword field
  - Define new messages to allow user to change the password protecting scenario classification level changes
  - Report Generator (on Server) will obtain classification level from the Scenario

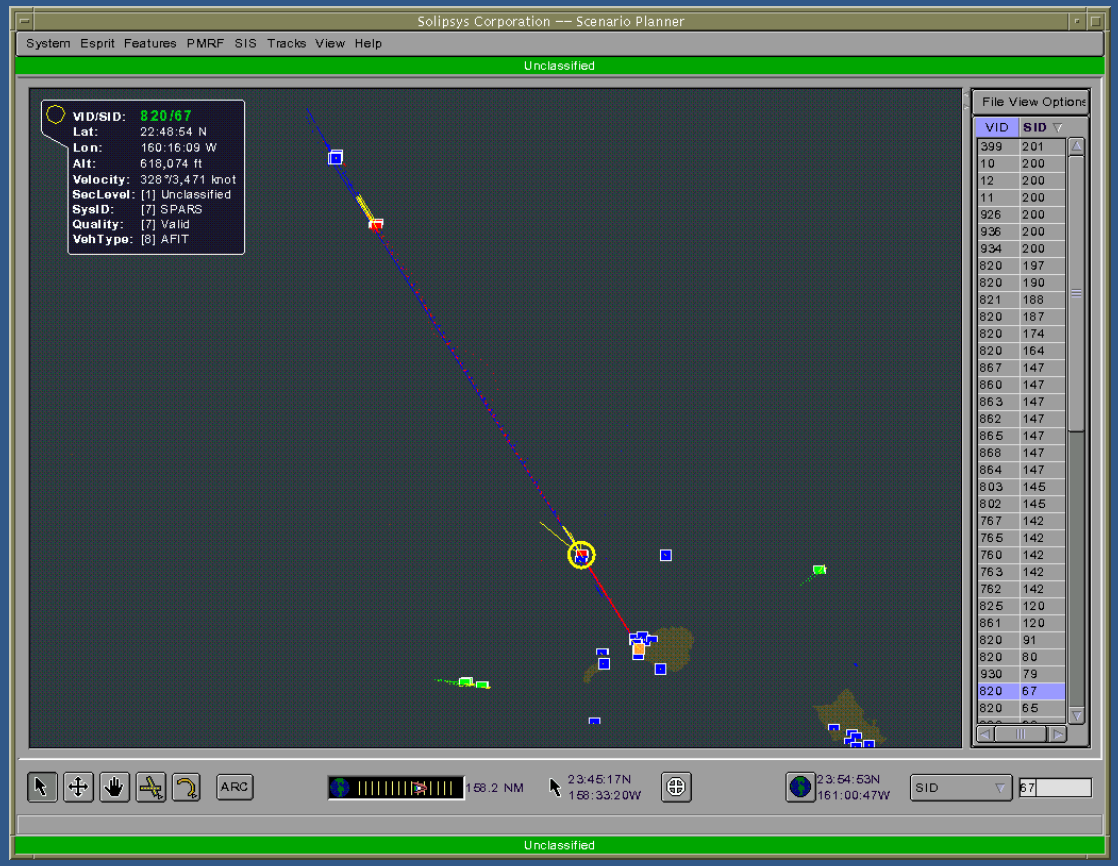
Classification Banner Design

- **Object Diagram**



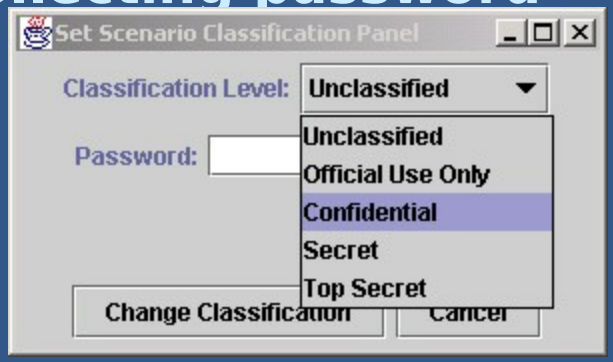
Classification Banner Design

- **GUI**
  - **TacSit with Classification Banner**



Classification Banner Design

- **GUI (Cont'd)**
  - **New screen to be added for changing scenario classification level and collecting password**



  - **Alert will notify user when classification banner changes due to receipt of live track message with security level higher than that currently displayed in classification banner**

Classification Banner Design

- **GUI (Cont'd)**
  - Classification Banner to be added to the following existing screens
    - Track List
    - Track Information
    - Alt Graph
    - Multigraph
  - Classification Banner to be added to the following reports
    - Waypoint Report
    - Time Interval Report
    - Planned Position Report
    - Sensor Report
    - New reports

      (Vehicle State Report, Vehicle Timeline Report, Scenario Timeline Report, Radar Assignment/ Coverage Report, Participant Location Map)

Classification Banner Design

- **GUI (Cont'd)**
  - **New screen to be added for changing scenario classification level and collecting password**
  - **New screen to be added for displaying alert to user when classification banner changes due to receipt of live track message with security level higher than that currently displayed in classification banner**
  - **New screen to be added to allow user to change the password protecting changes to a scenario's classification level**

Classification Banner Design

- **Build/Delivery Schedule**
  - **To be fully implemented for baseline functionality in February build**
  - **Classification banner will be added to new reports as they are included in subsequent builds**

Decision Aid Matrix

Decision Aid

- **Description**
  - **The user shall be able to define and enter operational constraints which will be monitored either automatically or manually during rehearsals or the actual event.**
  - **Upon selection of a constraint type, the user shall be presented with a list of predefined parameters required to completely express the constraint.**
  - **During the event, the automatically monitored constraints shall be evaluated on a periodic basis, and the status of each shall be presented in the Constraint Matrix.**

Decision Aid

- **Constraint Category**
  - Level of grouping for display of the Constraint Matrix
  - The user shall be prompted to enter a Category for each constraint.
  - Example Categories:
    - Aux Sensors
    - Pre-Launch(Target)
    - Pre-Launch (Intercept)
    - IP

Decision Aid

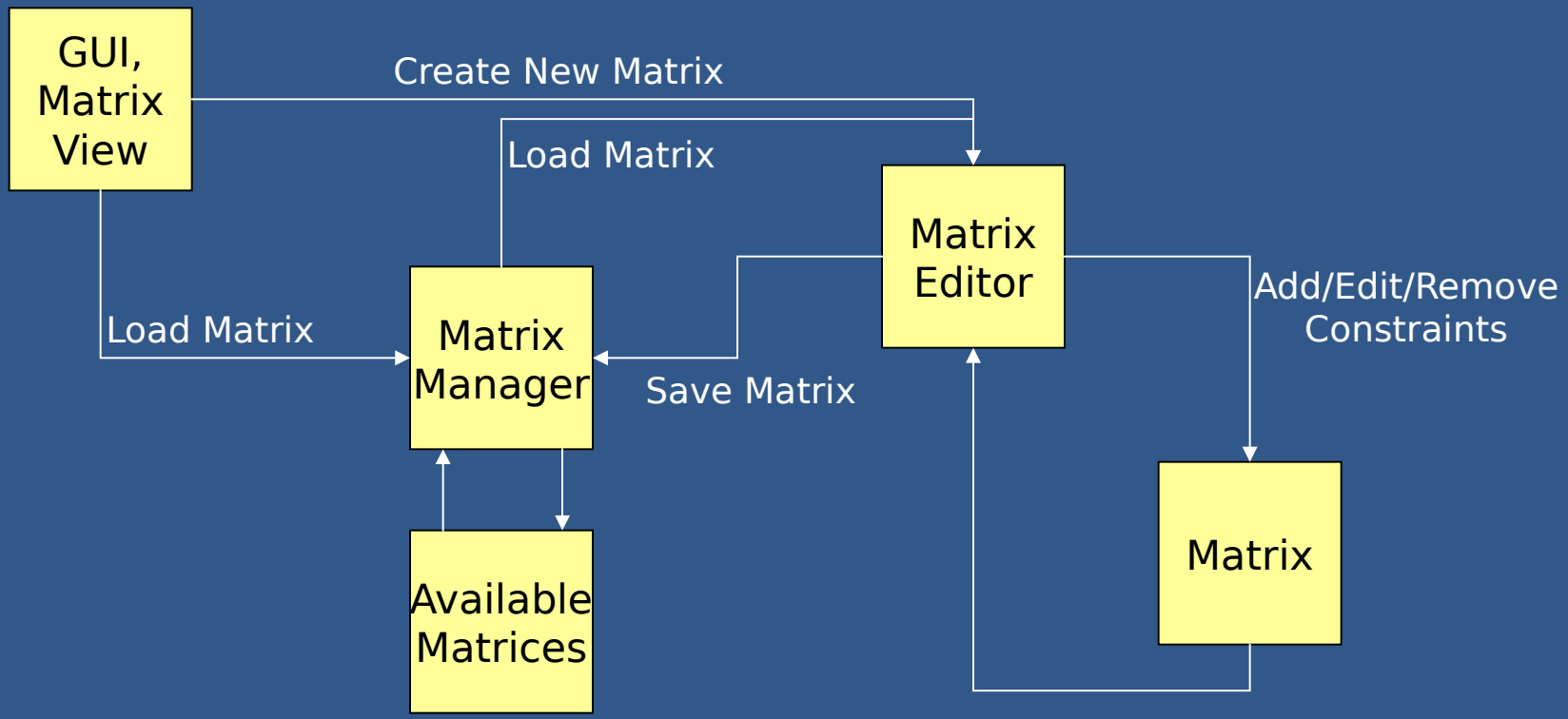
- **Constraint Type**
  - Distance
  - Manual
  - In The Box
  - Sensor On Line
  - Sensor On Target
  - Off Nominal (Distance)
  - Off Nominal (Time)
  - CPA

Constraint Violation Presentation

Type	Constraint Matrix	Symbology
Distance	Yes	Blinking Symbol
Manual	Yes	
In The Box	Yes	Blinking Symbol 
Sensor Online	Yes	Icon 
Sensor On Target	Yes	Icon
Off Nominal (Distance)	Yes	Blinking Symbol
Off Nominal (Time)	Yes	Blinking Symbol
CPA	Yes	Blinking Symbol

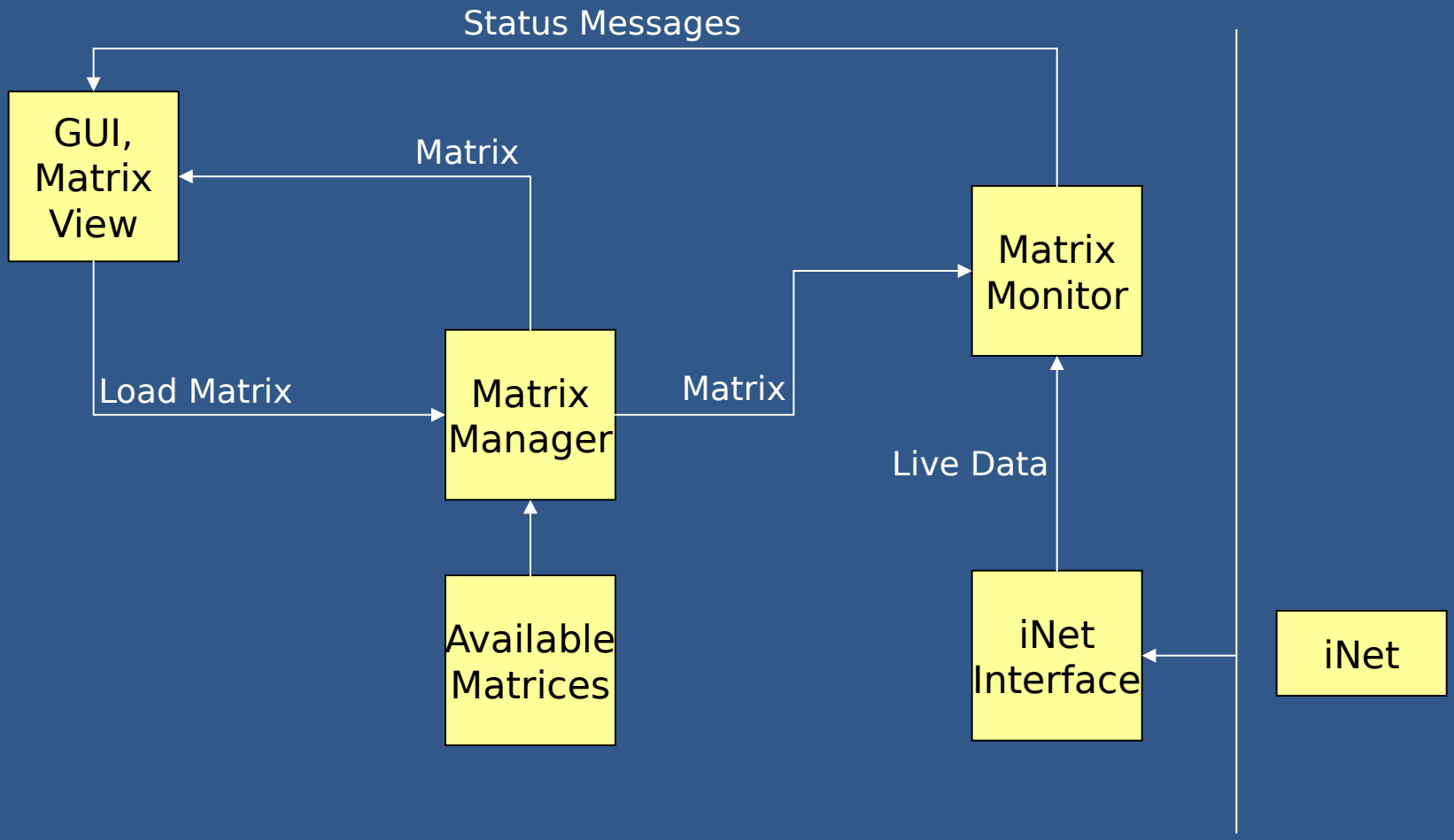
Decision Aid

- **Data Flow – Matrix Initialization**



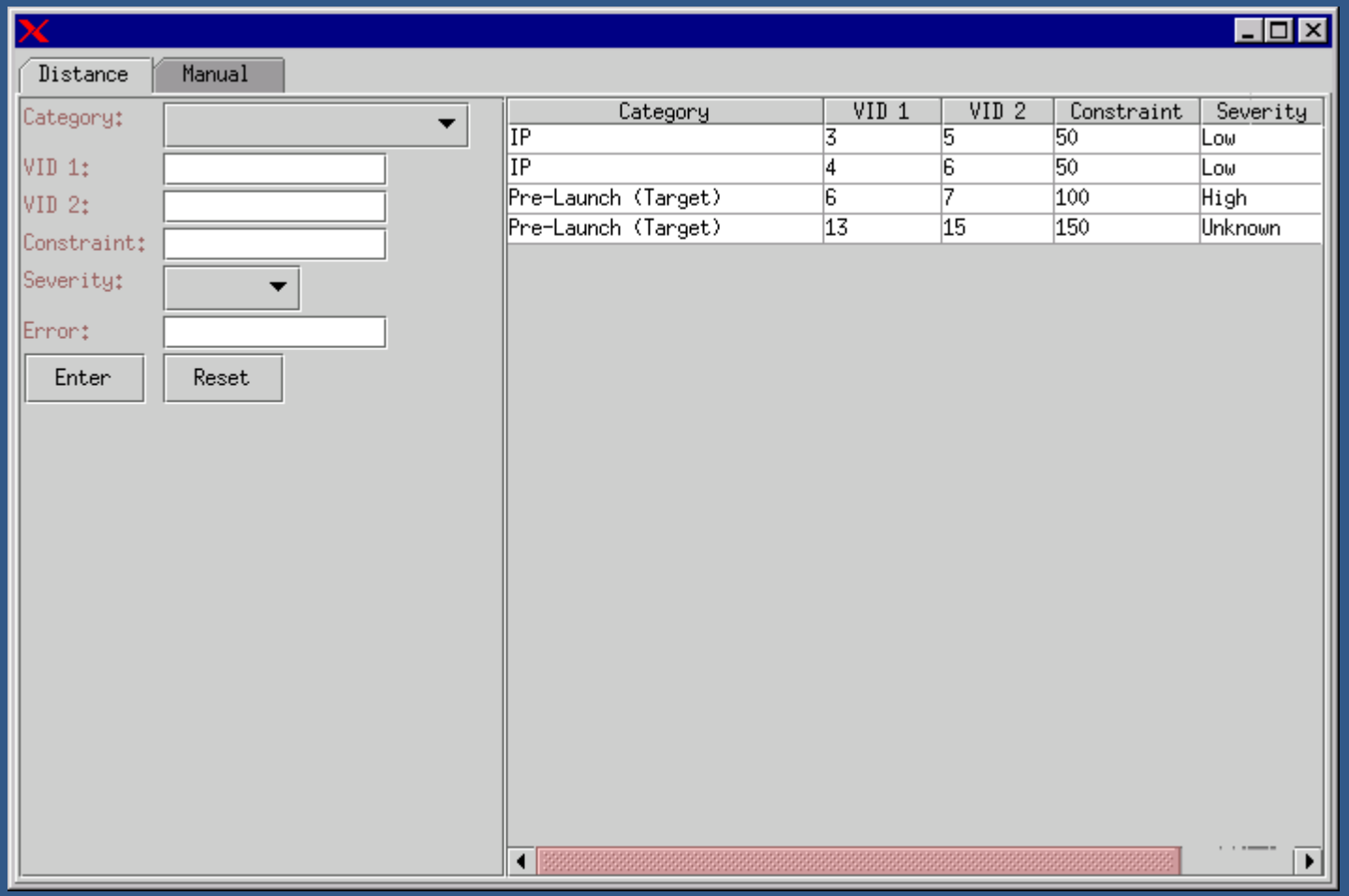
Decision Aid

- **Data Flow – Load/Activate Matrix**



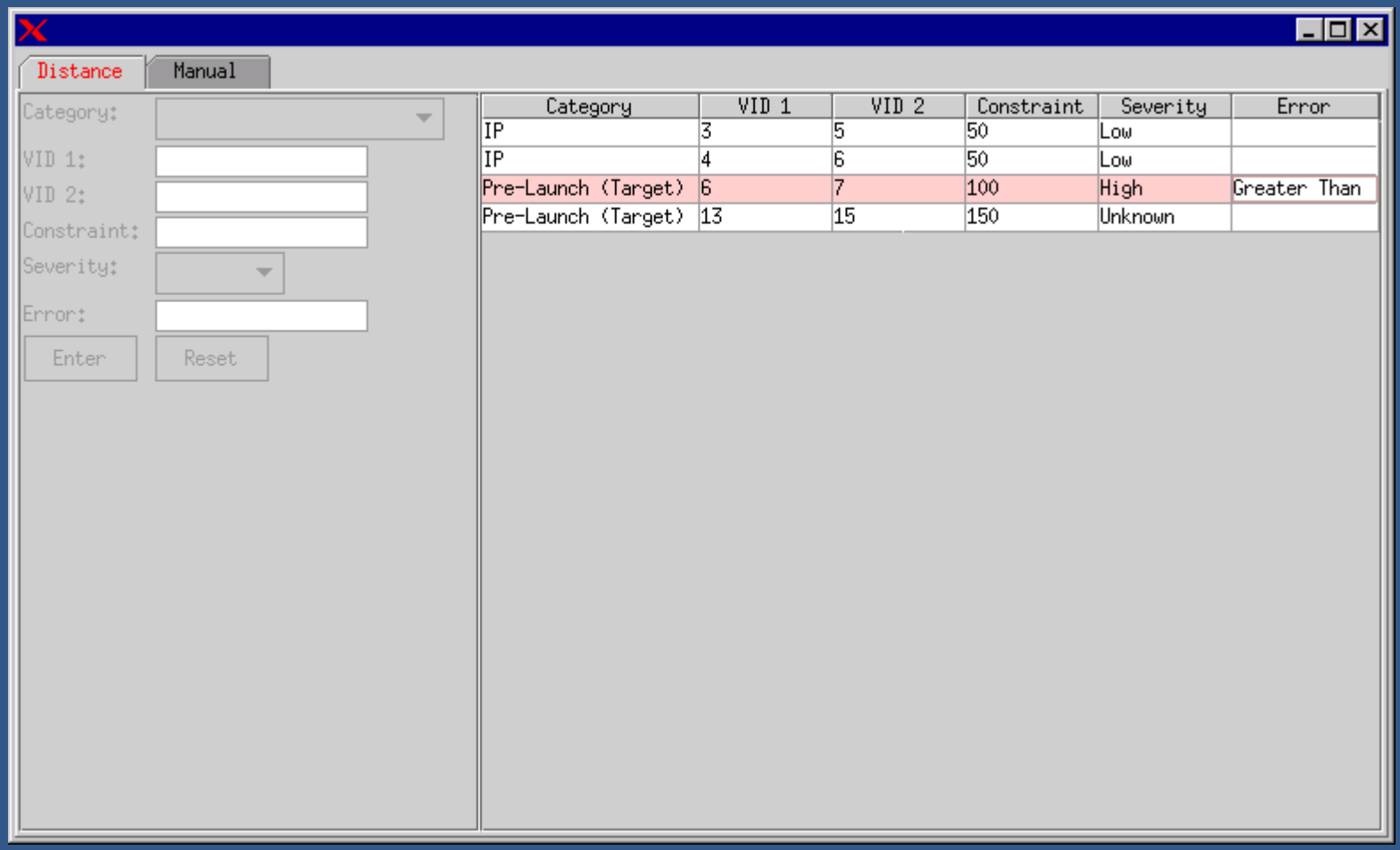
Decision Aid

- **GUI – Constraint Matrix Editor**



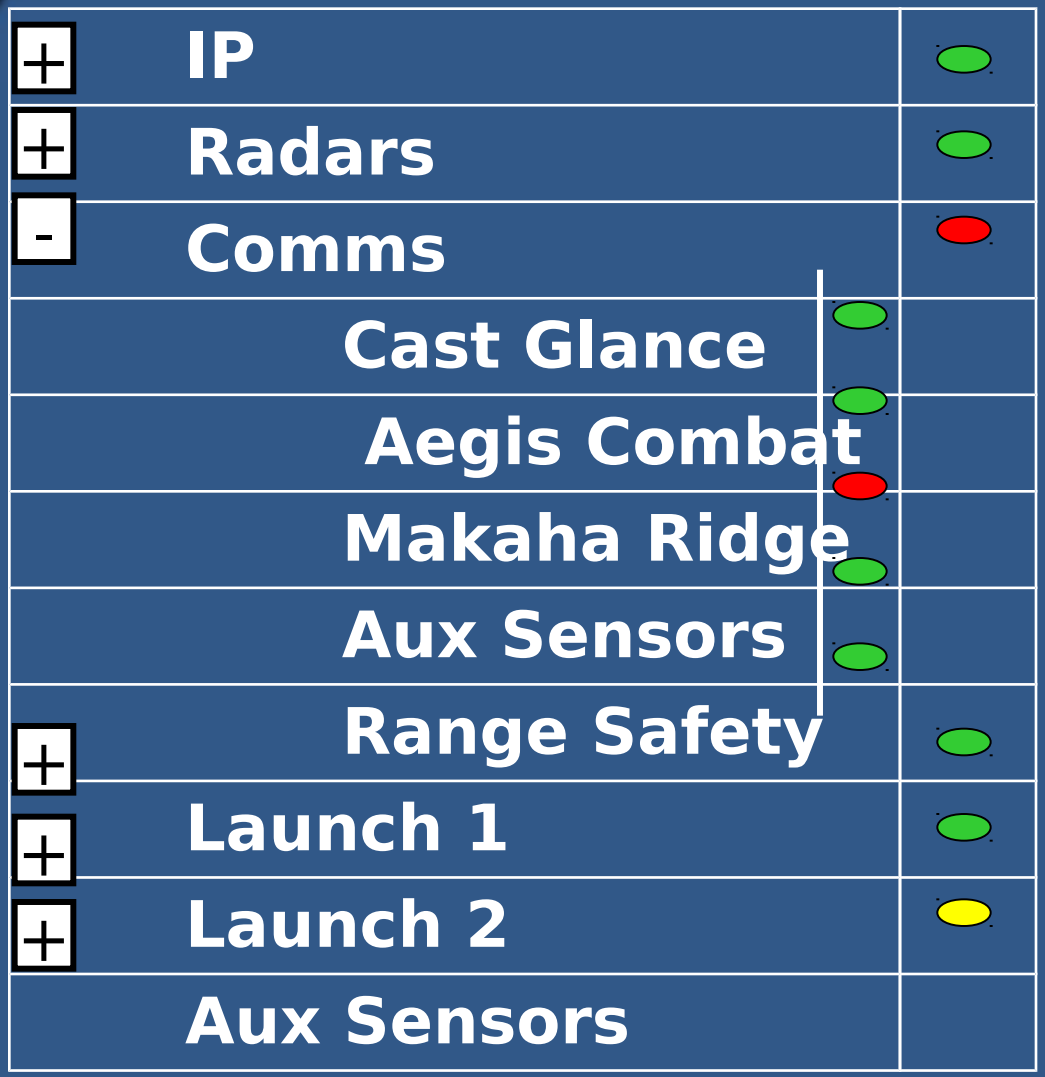
Decision Aid

- **GUI – Constraint Matrix View**



Decision Aid Matrix – Status Presentation

Decision Aid Matrix – Status Presentation



Decision Aid

- **Build Schedule**
  - **Build1 – 2/3/02**
    - **Constraint Entry Panel Complete**
    - **Constraint definition messages fully defined and implemented**
    - **Constraint monitoring algorithms defined and implementation completely defined**
    - **Minimal constraint monitoring implemented**
  - **Build 2 – 5/19/02**
    - **Constraint monitoring complete**
    - **Decision Aids complete**
  - **Build 3 – 9/1/02**
    - **Refinements**

Real-Time Data Presentation

- **Event Calculation & Display**

Event Calculation & Display

- **Real-time event calculations that have instantaneous numerical outputs are:**
  - **Launch Time**
  - **Apogee Altitude**
  - **Apogee Time**
  - **IIP Altitude**
  - **IIP Time**
  - **Splash Time**
  - **Closest Point of Approach (CPA)**
  - **CPA Time OR Time-to-go**

Event Calculation & Display

Instantaneous Real-time Events

- These calculations are automatically performed on the appropriate vehicles and may be performed on other vehicles as specified by the GUI interface

- The results will be displayed in several ways:

  - Columns in the track list
  - Track Tags
  - Track Details Box

Event Calculation & Display

- **Launch Time Calculation**
  - Radar measurements are processed
  - Time of first statistically significant motion is estimated using a multiple hypothesis batch estimator
- **Instantaneous Impact Prediction**
  - A gravitational ballistic prediction is performed using a variable step-size numerical integrator
  - Used to determine apogee, apogee time, altitude, altitude time, and splash
  - Used to determine CPA for ballistic objects
- **CPA Calculation**
  - Air Targets – straight line motion is assumed for CPA calculation
  - Both ground range CPA and slant range CPA will be computed and displayed

Event Calculation & Display
Build Schedule

- **Build 1 – 2/3/02**
  - Presentation designed and implemented
  - Event calculations designed
  - Messages defined
- **Build 2 – 5/19/02**
  - Implementation complete
- **Build 3 – 9/1/02**
  - Refinements

Mobile Sensors

Mobile Sensors

Combined with Radar Coverage Analysis

Scenario Plan Manipulation

Mission Planning

- **Rotation**
  - User inputs origin and magnitude of rotation through GUI
  - All associated points are rotated about the the axis normal to the origin's local tangent plane
  - The rotation moves waypoints along a geodesic:
    - Orthogonal to the line of sight from the origin
    - A distance consistent with the specified change in azimuth

- **NOTE: Large rotations and translation can cause waypoint distortion – User can elect to preserve ground range separation and altitude of waypoints**

Mission Planning

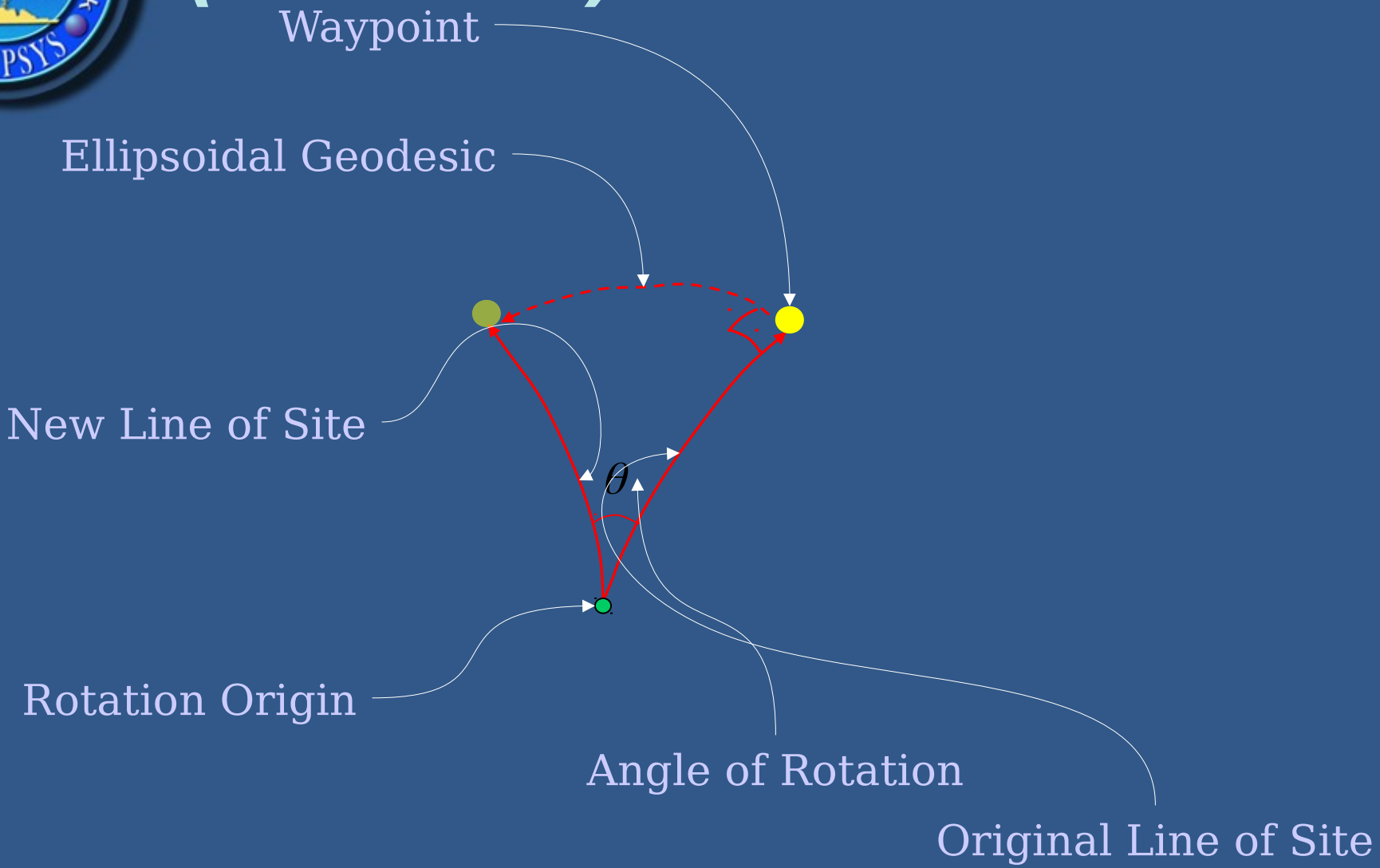
- **Translation**
  - **User can drag and drop OR manually enter north-east offset**
  - **All points are translated along a geodesic path by:**
    - **Moving the points in the direction indicated by the offset**
    - **Moving the points a distance indicated by the offset magnitude**

- **NOTE: Large rotations and translation can cause waypoint distortion – User can elect to preserve ground range separation and altitude of waypoints**

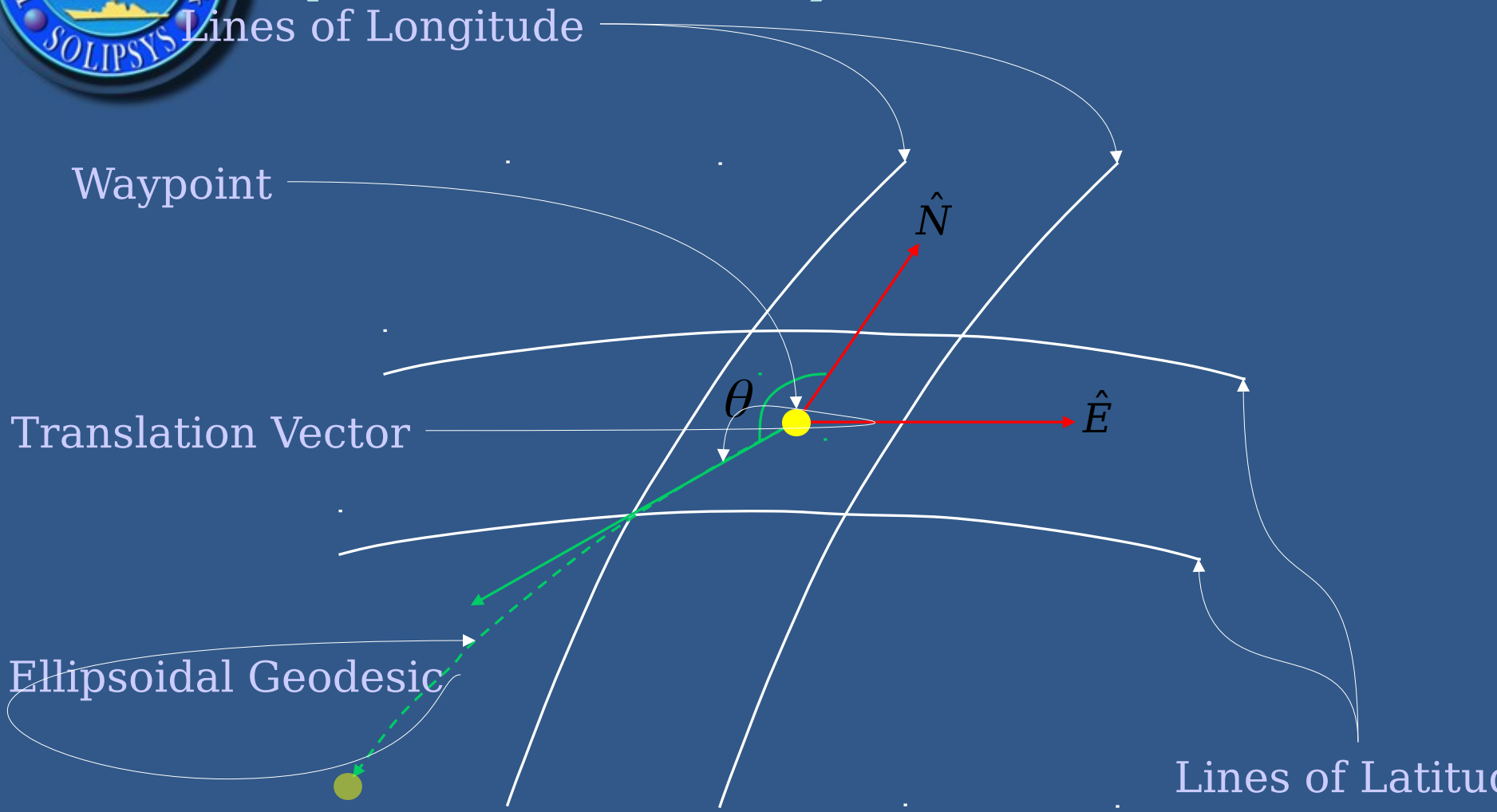
Mission Planning

- **Mirror Flip**
  - **User specifies an origin and azimuth OR two geodetic points**
  - **Waypoints are “rotated” to the other side of this tangent line**

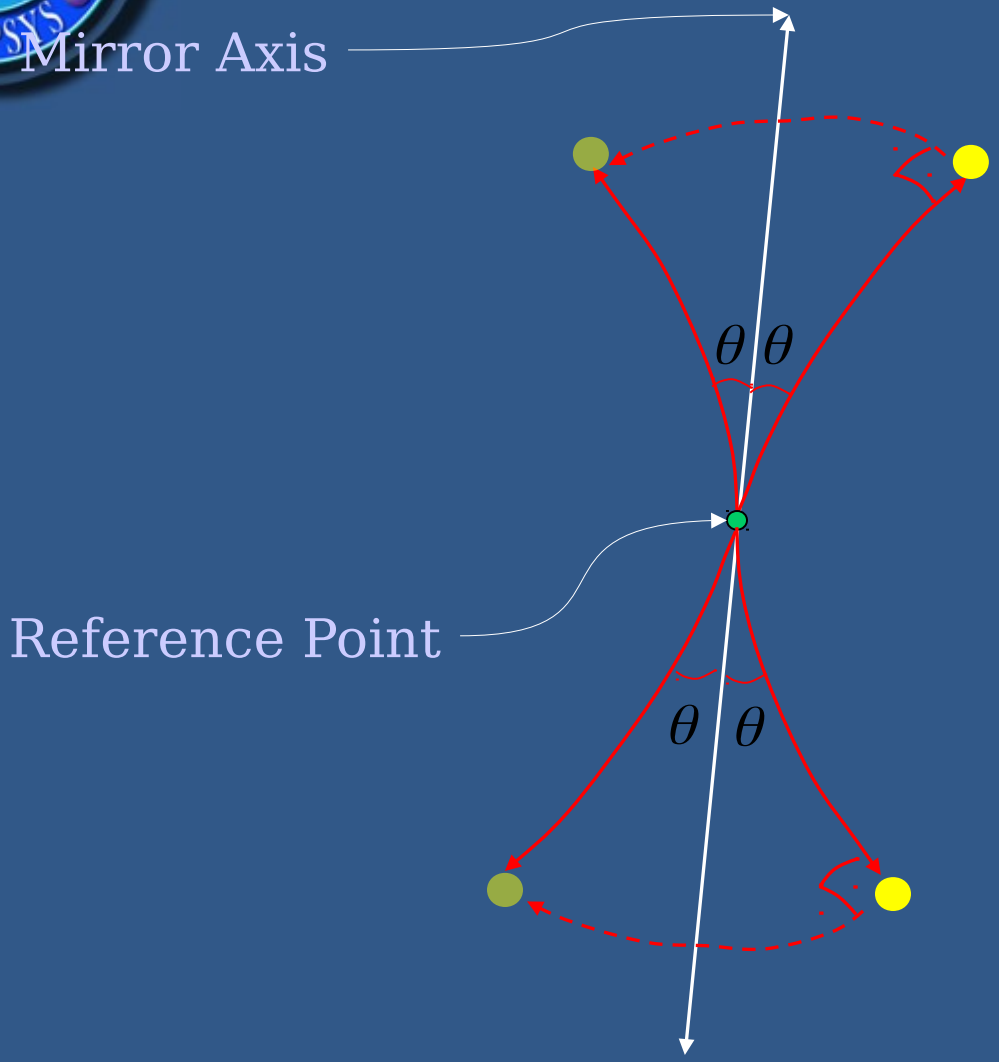
Mission Planning (Rotation)



Mission Planning (Translation)



Mission Planning (Mirror Flip)



Radar Coverage Analysis

Modes of ESPRIT - Mission Planning

Side bar text.

- **Mission Planning:**
  - Sensor models are defined
    - Mode parameters
  - Sensor entities are defined
    - Stationary
    - Mobile
    - Shadow masks
    - Cut-outs
    - Vehicle reporting responsibility
  - Vehicles are defined
    - Waypoints
    - Timeline
    - RCS
  - Vehicles coverage analysis is performed
    - S/N analysis
    - Coverage summary

Modes of ESPRIT - Mission Preview

Side bar text.

- **Mission Preview**
  - **Graphical illustration of line-of-sight is indicated for each radar-target pairing**
  - **User alerted if a target completely loses coverage**
  - **Vehicles icons are displayed along with planned trajectory**

Modes of ESPRIT- Mission Rehearsal

Side bar text.

- **Mission Rehearsal**
  - **Msg25 and/or Msg26 data can be distributed on the iNet**
  - **Track icons appear when sensors are “on track”**
  - **iNet data is simulated using sensor parameters and real-time data processing**

Mission Planning (Sensors)

Side bar text.

- **Pointing Radars**
  - Each sensor model has configuration parameters including, but are not limited to:
    - Maximum detection range
    - Angular slew rates
    - Transmitter Power
    - S/N Threshold
    - RF Loop Gain
    - Beamwidth
  - A sensor model is unique to a sensor type but independent of sensor location and vehicle assignment
- **Fixed Radars**
  - Angular slew rates are irrelevant
  - Beam width is replaced by half-angle

Mission Planning (Sensors)

Side bar text.

- **Sensor Entities**
  - Stationary
    - Shadow masks – due to land masses, buildings, etc..
    - Radiation hazard cut-outs – due to EMI considerations, etc…
    - Boresight (Fixed radar only)
  - Mobile
    - Shadow masks, Radiation hazard cut-outs, and Boresight data are relative to platform heading
  - Coverage zones are input using comma delimited ASCII text format
- **ASCII text description**
  - An azimuth and elevation range is specified by comma delimited text
  - This information is combined with sensor model parameters to compute radar coverage

Radar Coverage Analysis – Mission Planning (Sensors)

Load Sensor Specific Data

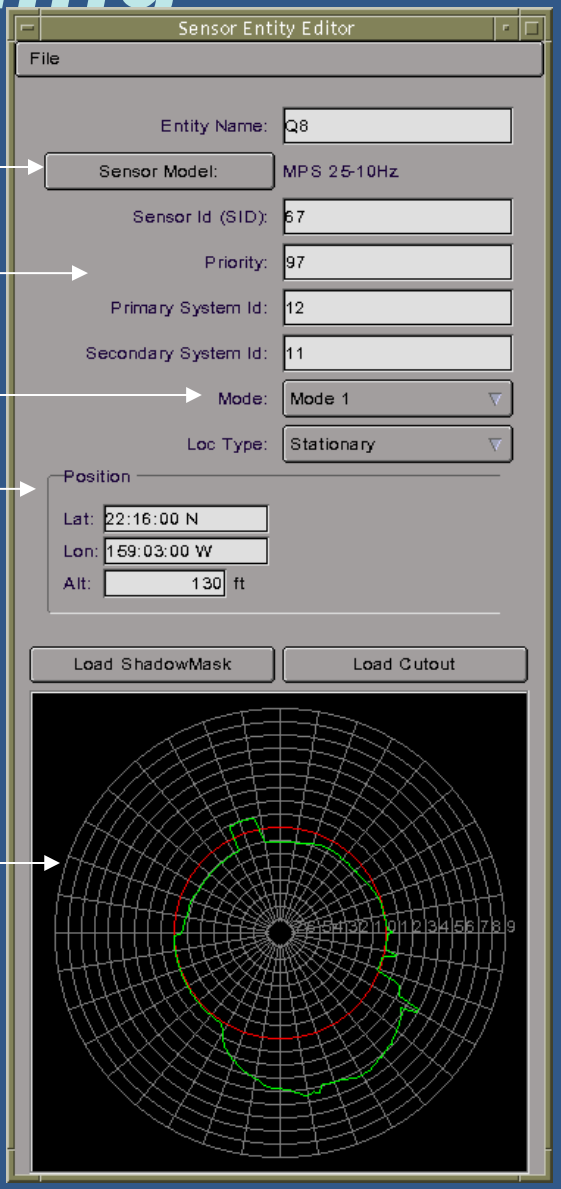
Side bar text.

Enter Site Parameters

Select Sensor Operation Mode

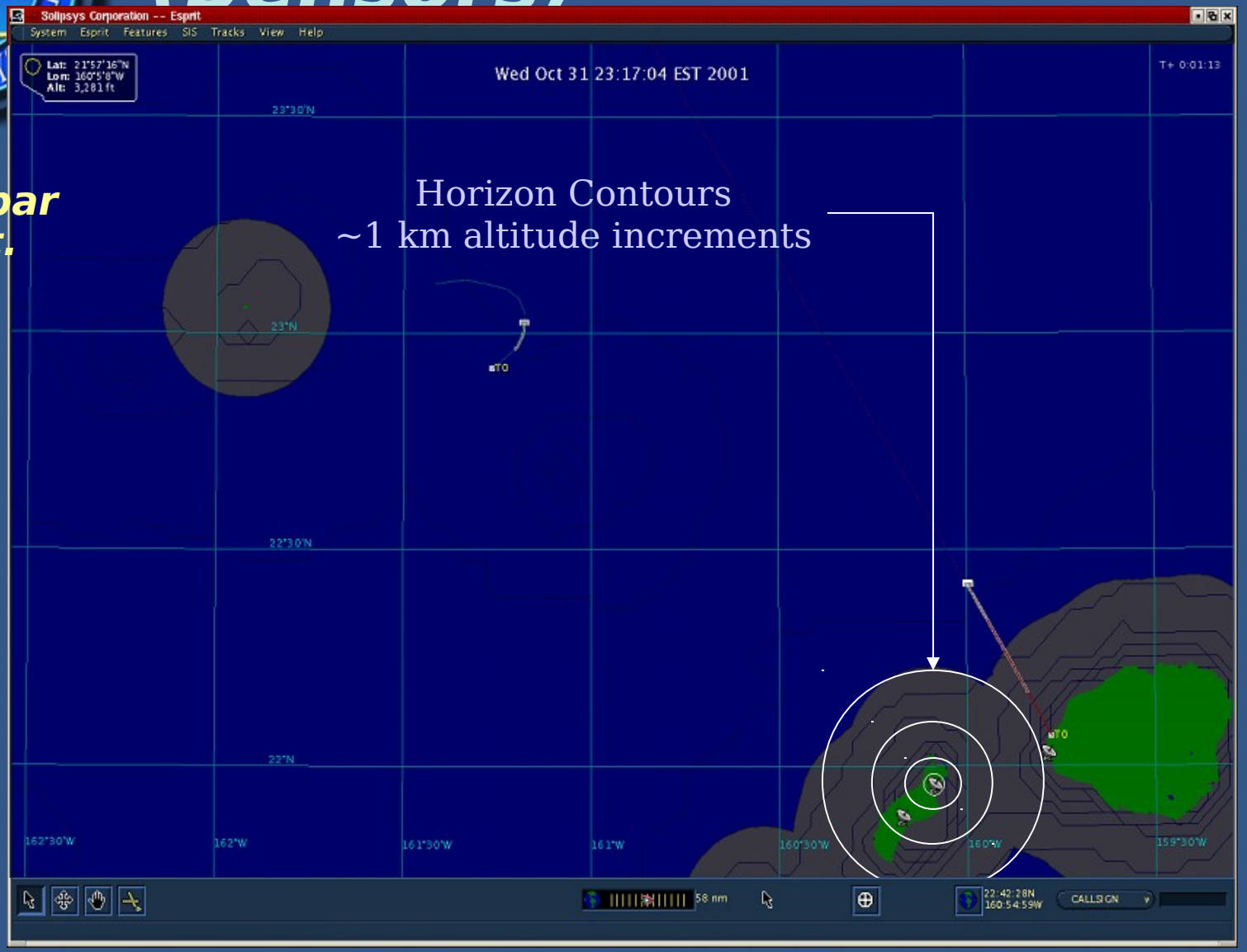
Stationary or Mobile

Shadow Mask and Cut-out Plot



Mission Planning (Sensors)

Side bar text.



Mission Planning (Vehicles)

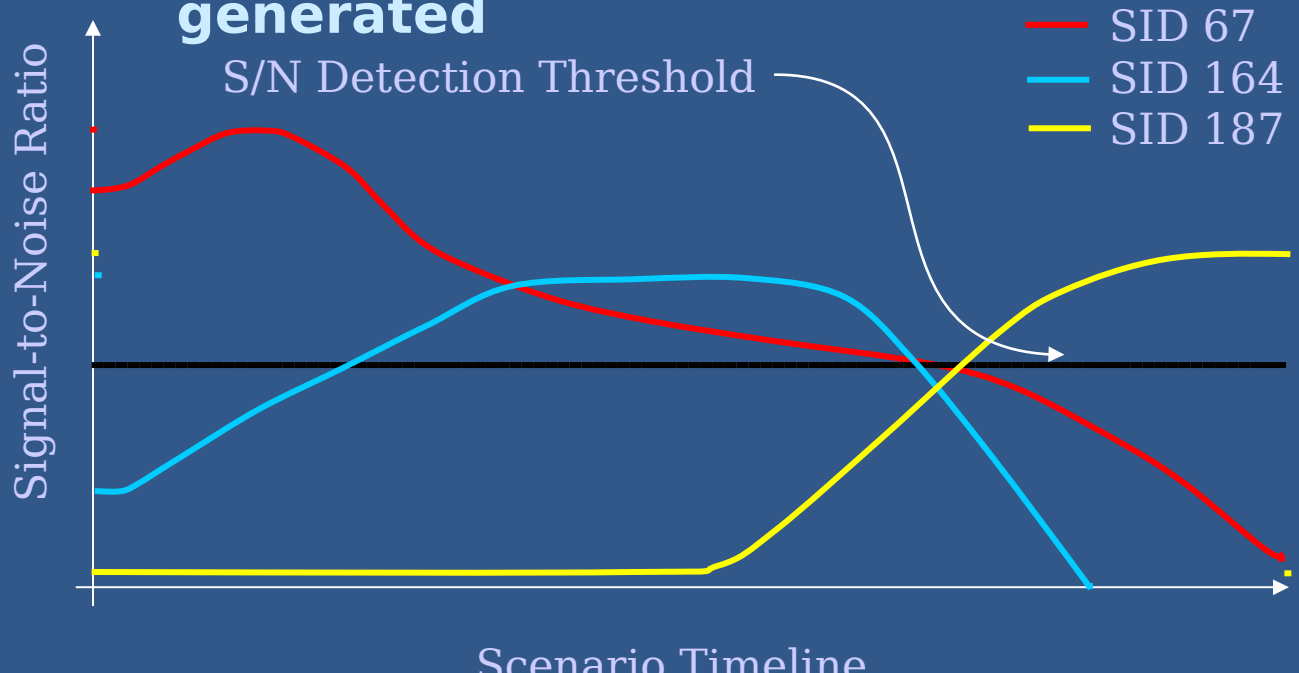
Side bar text.

- **Vehicle waypoints can be specified in the following ways:**
  - **Geodetic location and time-of-arrival**
  - **Geodetic location and instantaneous speed**
  - **Geodetic location and instantaneous acceleration – linear or coordinated turn**
  - **Geodetic location and trajectory type – circular or racetrack**
- **Vehicle trajectories can be synchronized:**
  - **User selects multiple vehicle trajectories**
  - **User selects “Synchronize” from the ESPRIT menu**
  - **User enters synchronization time**
- **Vehicle RCS is defined:**
  - **RCS is provided as a function of aspect angle**
  - **This can be done by comma delimited text similar to shadow mask definitions**

Mission Planning (Vehicles)

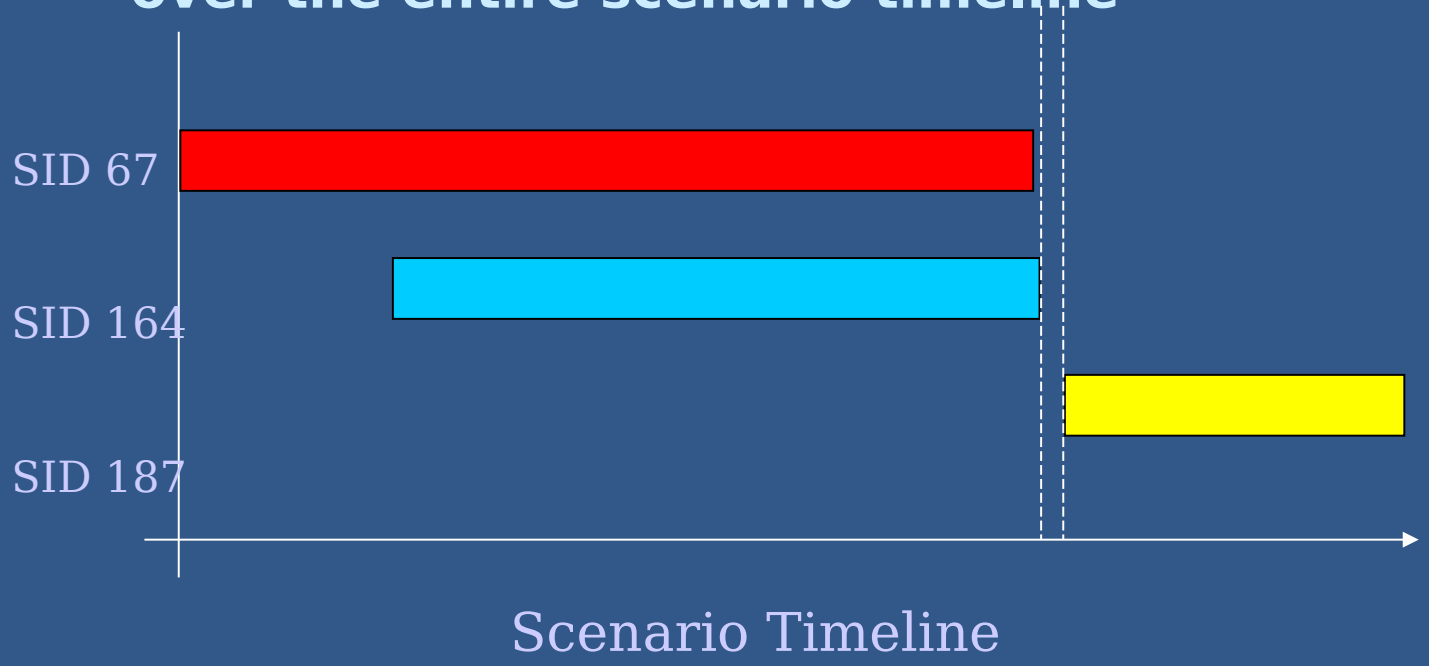
Side bar text.

- **S/N Plots:**
  - **A VID is selected by the user**
  - **The S/N Plot option is selected**
  - **Using sensor model, sensor entity, vehicle waypoints, and vehicle RCS, a S/N graph is generated**



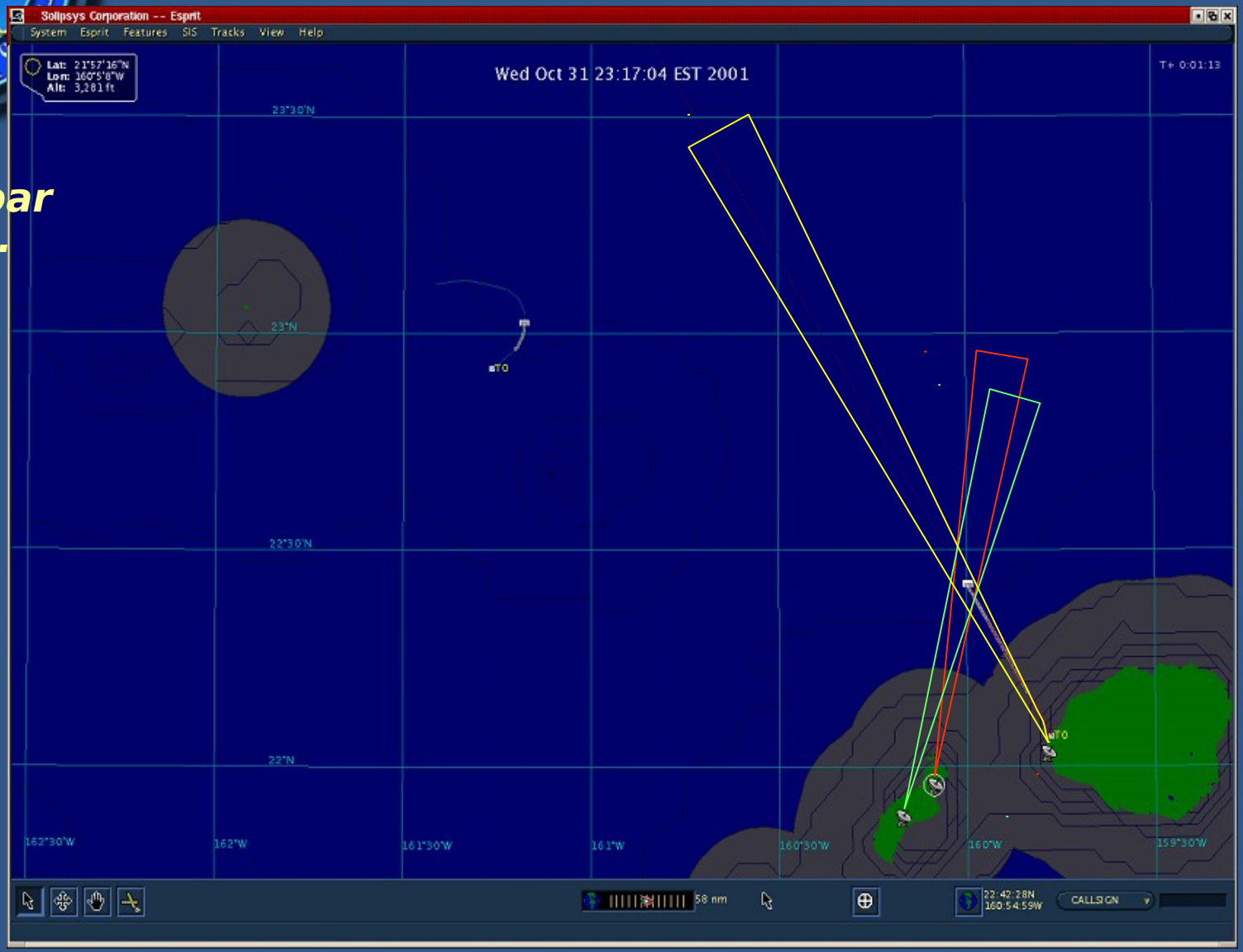
Mission Planning (Vehicle)

- **Vehicle Coverage Reports:**
  - **User selects VID for radar coverage report**
  - **Using S/N data for this VID, a coverage report is generated for the selected vehicle over the entire scenario timeline**



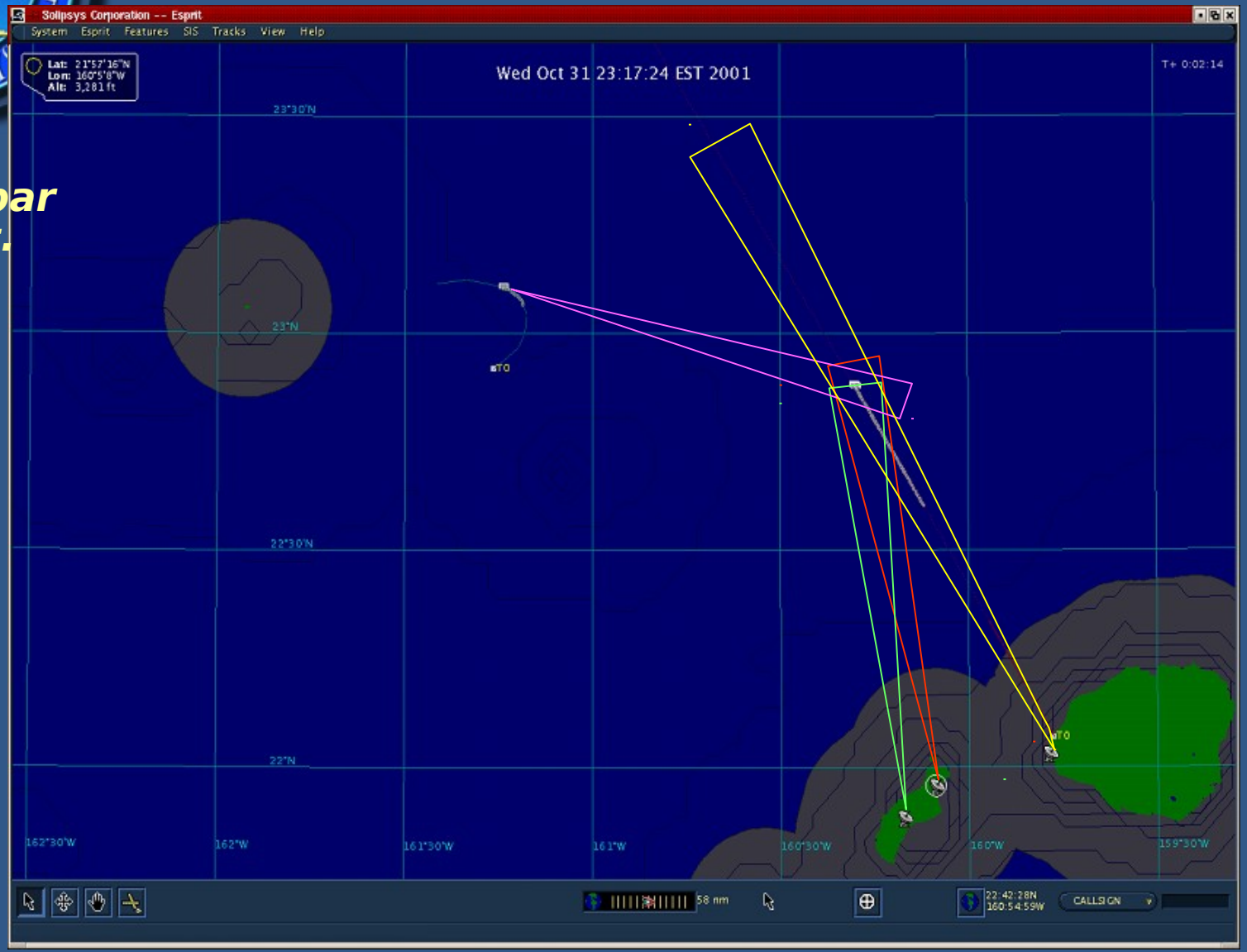
Mission Preview

Side bar text.



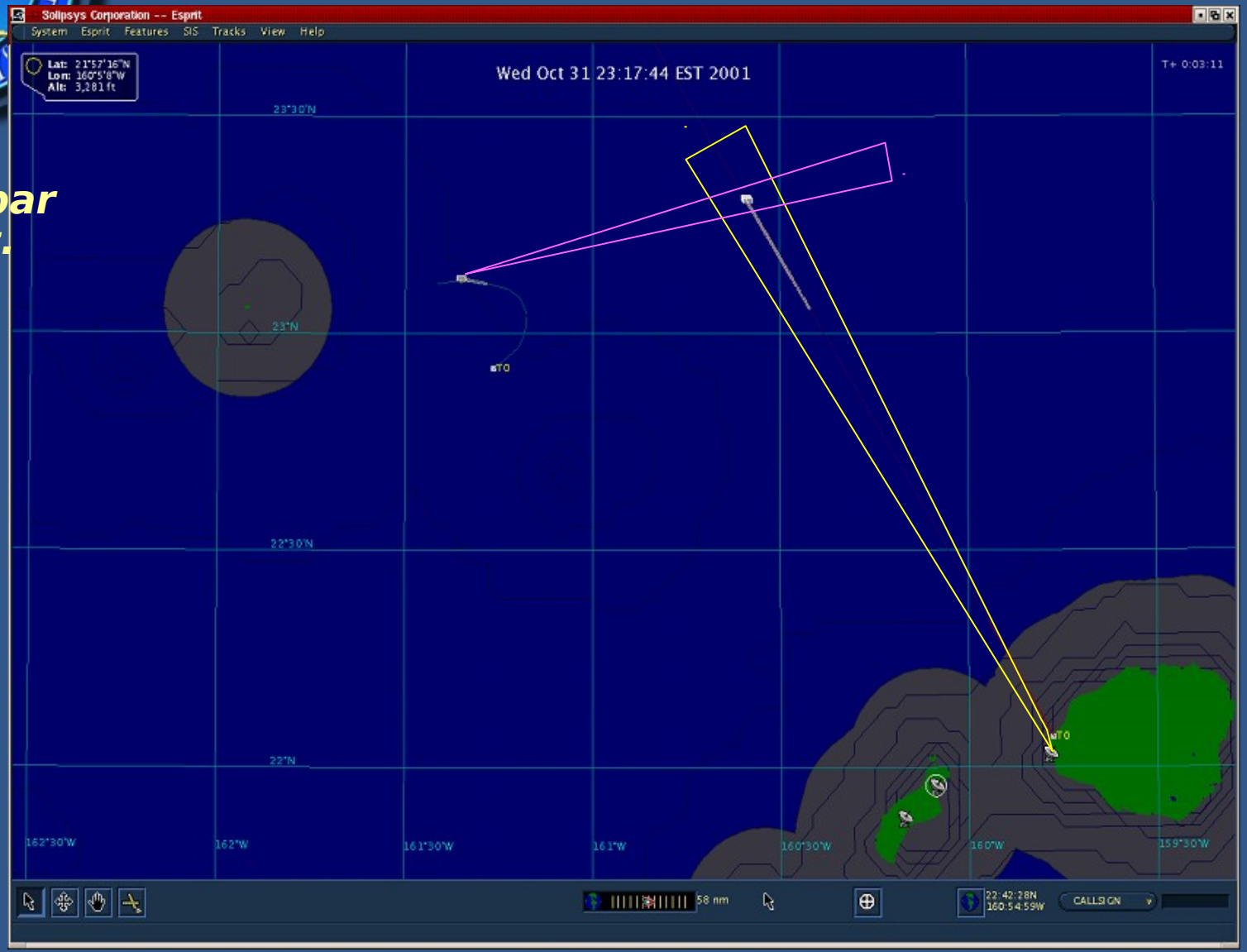
Mission Preview

Side bar text.



Mission Preview

Side bar text.



Plan Configuration Management

Scenario Configuration Management

- **Working version**
  - Used during initial scenario design
  - Experimentation encouraged
- **Certified versions**
  - Once scenario design work is complete, scenario is certified
  - Non-secure password may be employed to restrict certification permission
  - Multiple versions of certified scenario need to be saved/accessible

Scenario Configuration Management

- CVS backend
  - Allows comments to be saved with each certification of a scenario
  - User ID is logged with change
  - Any past versions of a scenario are easily recovered
  - Allows flexible options
    - Shared access from multiple networked sites
    - Web front-end to CVS

Scenario Configuration Management

Certify scenario

Scenario Editor

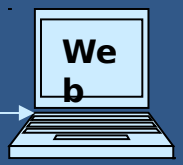
Logon Dialog

NO

Is User Logged On?

CANCEL

YES



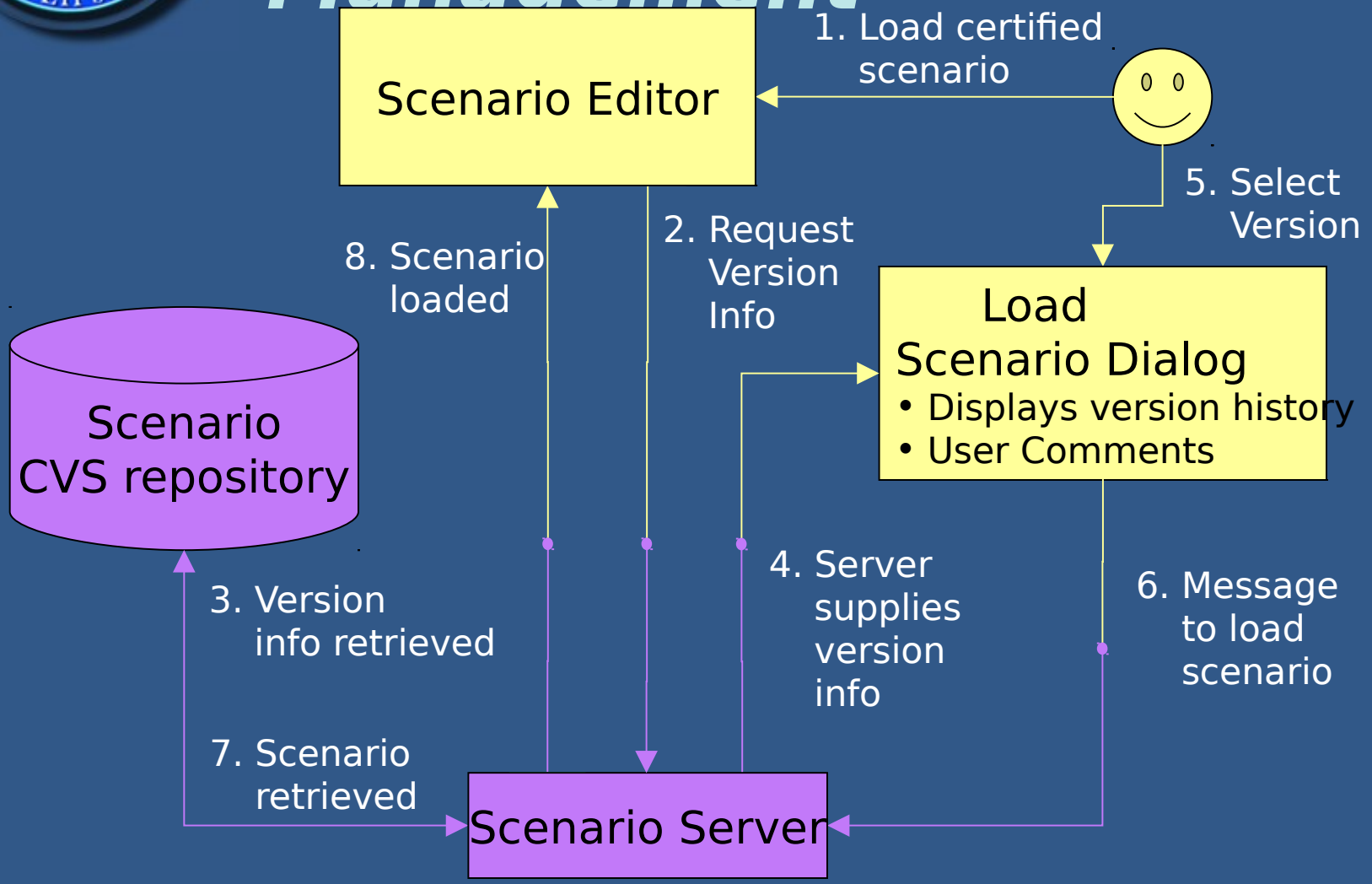
CVS repository

Certify Scenario Dialog
- Scenario Name
- User Comments

Scenario Server

Serialized scenario

Scenario Configuration Management



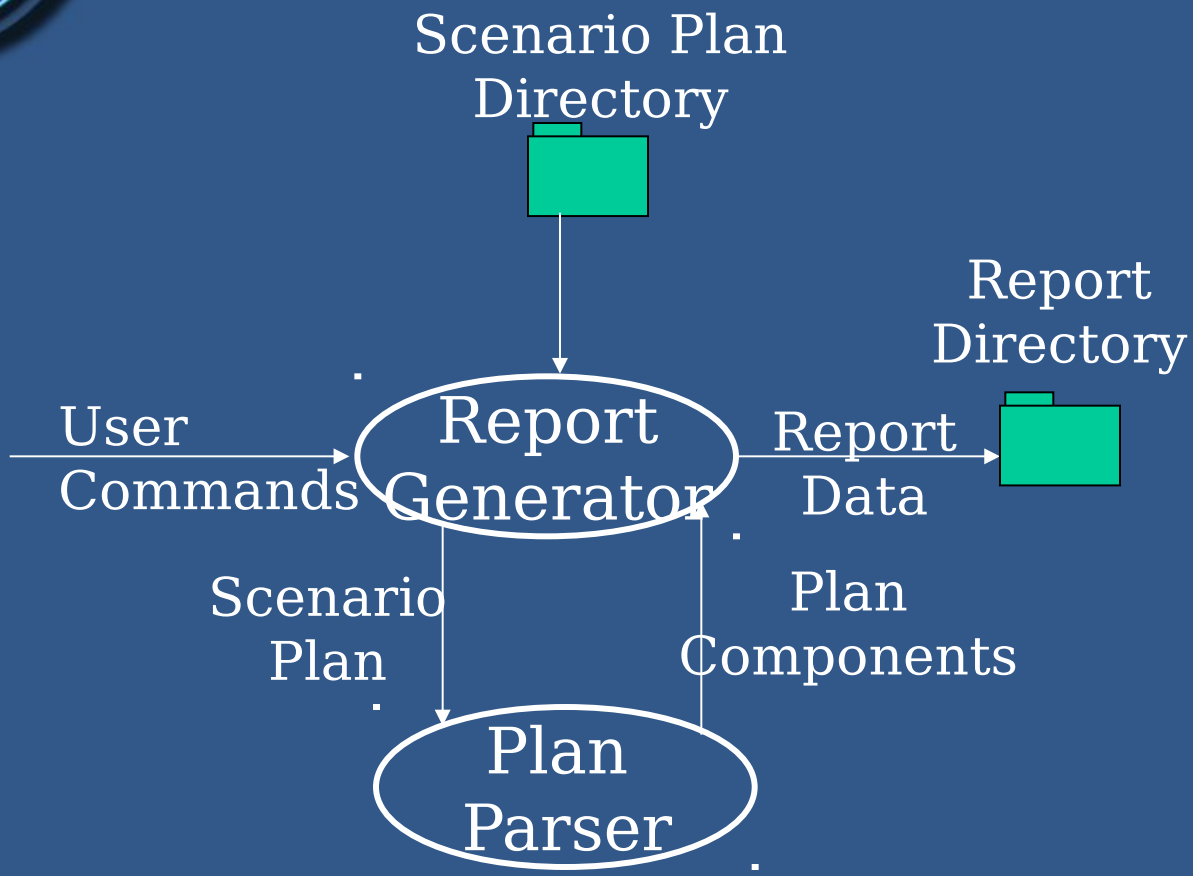
Plan Reports

Planning Reports

Plan Reports Assure Consistency And Concurrency Between the Plan and what is published

- **The Mission Plan is the repository of a considerable amount of mission information**
  - **Vehicle State Report**
  - **Vehicle Timeline Report**
  - **Scenario Timeline Report**
  - **Radar Coverage Report**
  - **Vehicle Information Table (VIT)**
  - **Sensor Information Table (SIT)**

Planning Reports: Data Flow Diagram



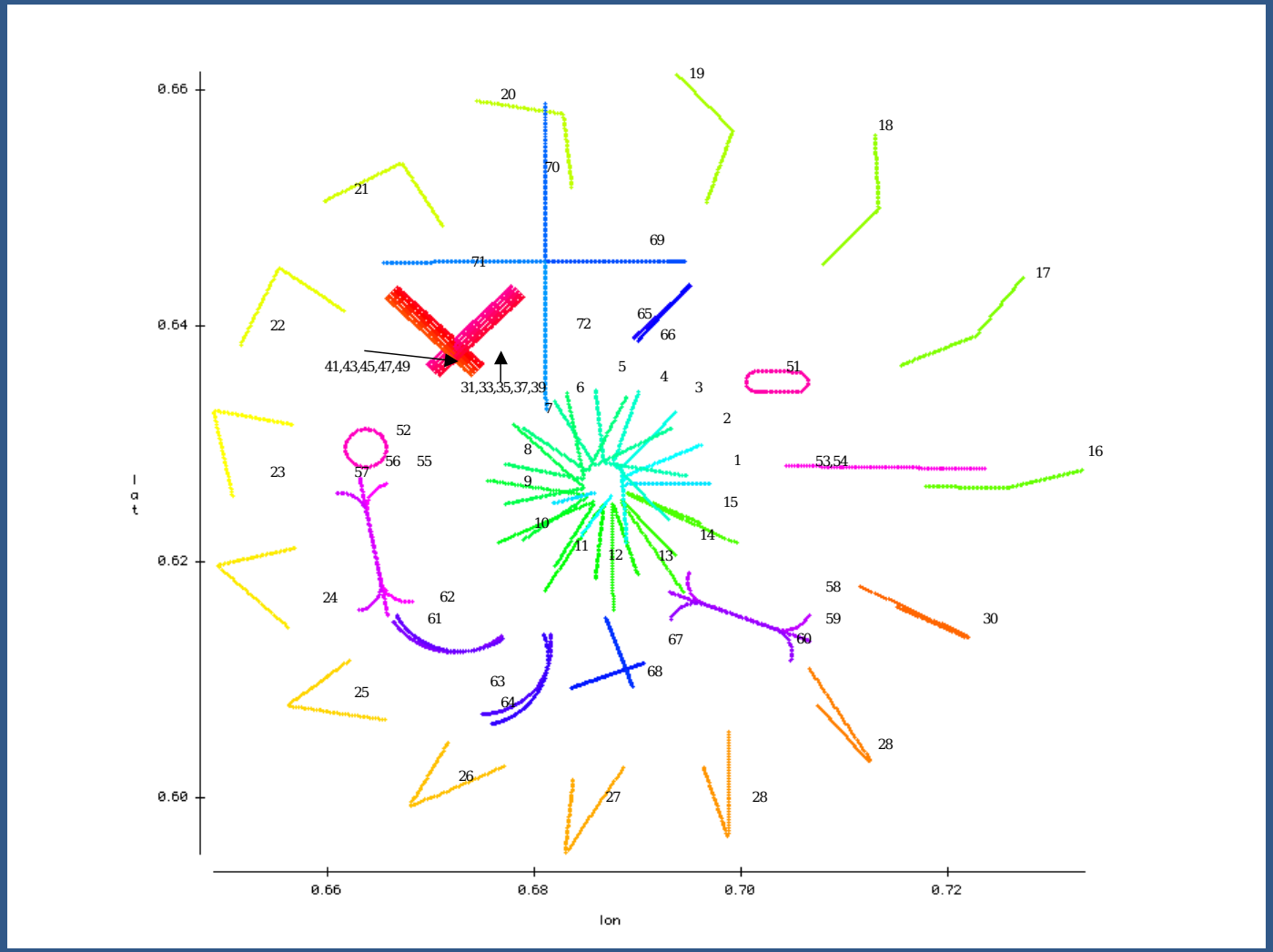
Vehicle State Report

The Vehicle State report provides time, position, velocity and other application pertinent data.

4300 50 1 -89393.15853106 65305.32627878 0.00000000 106.06601718 106.06601718 0.00000000 11
.94683067 0.02301077 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.000000
9350 50 1 -88857.52514431 65840.95966553 0.00000000 106.06601718 106.06601718 0.00000000 11
.93785813 0.01382696 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.000000
14450 50 1 -88316.58845670 66381.89635314 0.00000000 106.06601718 106.06601718 0.00000000 1
0.92467508 0.01830606 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
19500 50 1 -87780.95506995 66917.52973989 0.00000000 106.06601718 106.06601718 0.00000000 1
0.92082443 0.01793941 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
24600 50 1 -87240.01838234 67458.46642750 0.00000000 106.06601718 106.06601718 0.00000000 1
0.90692996 0.02354703 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
29650 50 1 -86704.38499559 67994.09981425 0.00000000 106.06601718 106.06601718 0.00000000 1
0.90918118 0.02992294 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
34700 50 1 -86168.75160884 68529.73320100 0.00000000 106.06601718 106.06601718 0.00000000 1
0.90368232 0.01818685 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
39800 50 1 -85627.81492124 69070.66988860 0.00000000 106.06601718 106.06601718 0.00000000 1
0.89617175 0.02458738 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
44850 50 1 -85092.18153449 69606.30327535 0.00000000 106.06601718 106.06601718 0.00000000 1
0.88719894 0.03167205 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000
49900 50 1 -84556.54814774 70141.93666210 0.00000000 106.06601718 106.06601718 0.00000000 1
0.87548217 0.01837765 140.00000000 0.00500000 0.00500000 1 0.62667016 2.25781610 1524.00000

Application of Vehicle State Report

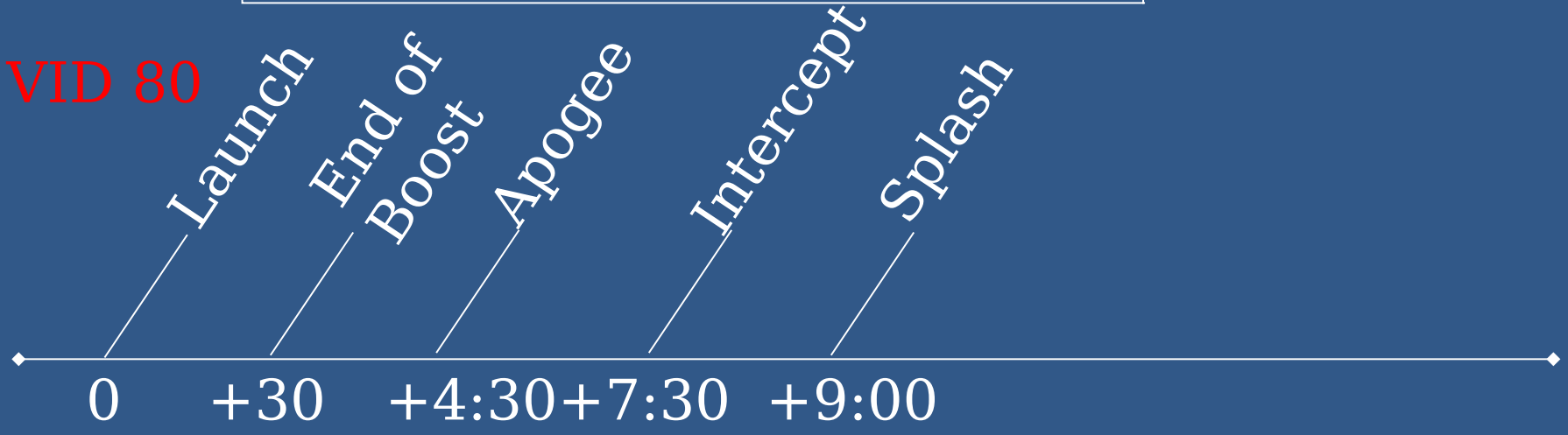
Solipsys application is to use the planning tool to generate realistic test cases for presentation to tracking systems



Vehicle Timeline Report

VID 80

- T+0 Launch
- T+30 End of Boost
- T+4:30 Apogee
- T+7:30 Intercept
- T+9:00 Splash

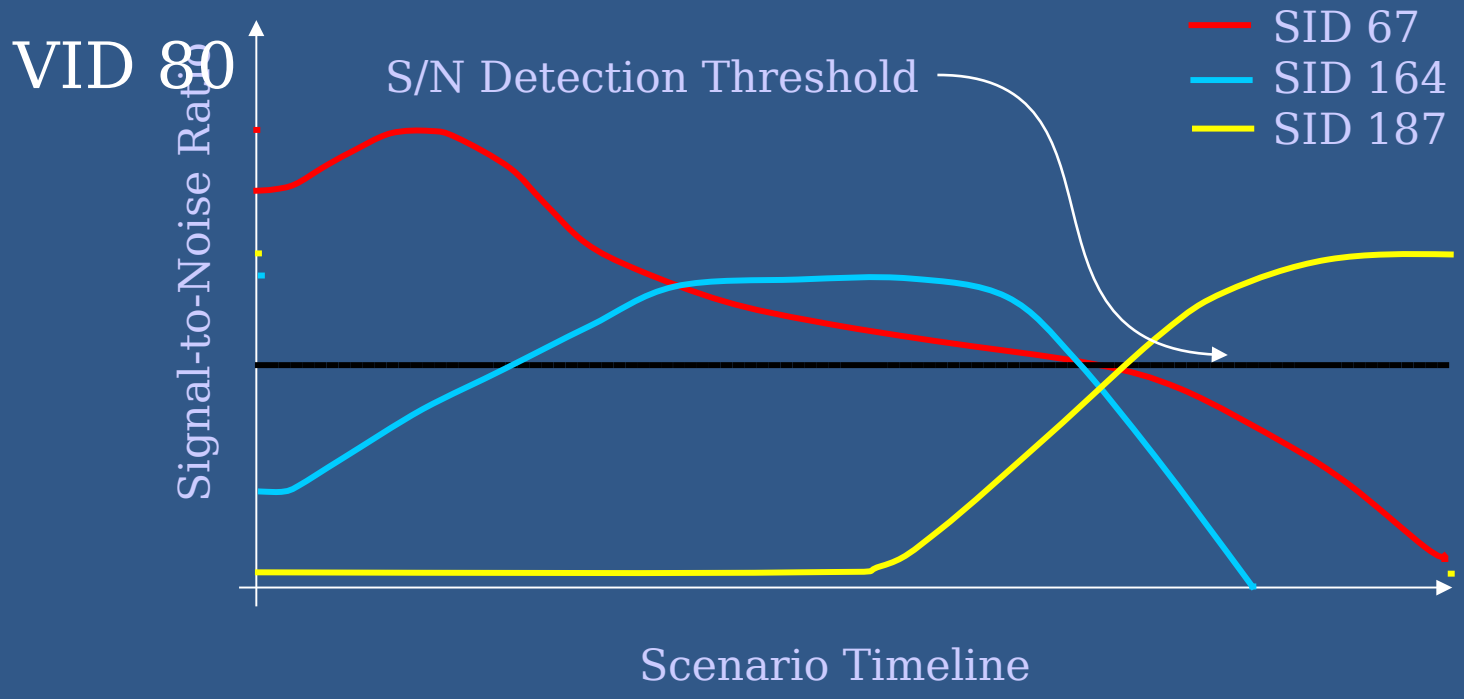


Scenario Timeline Report

VID	Time	Description	
80	T+0	Launch	
80	+4:30	Apogee	
90	+5:36	Launch	
90	+7:30	Intercept	

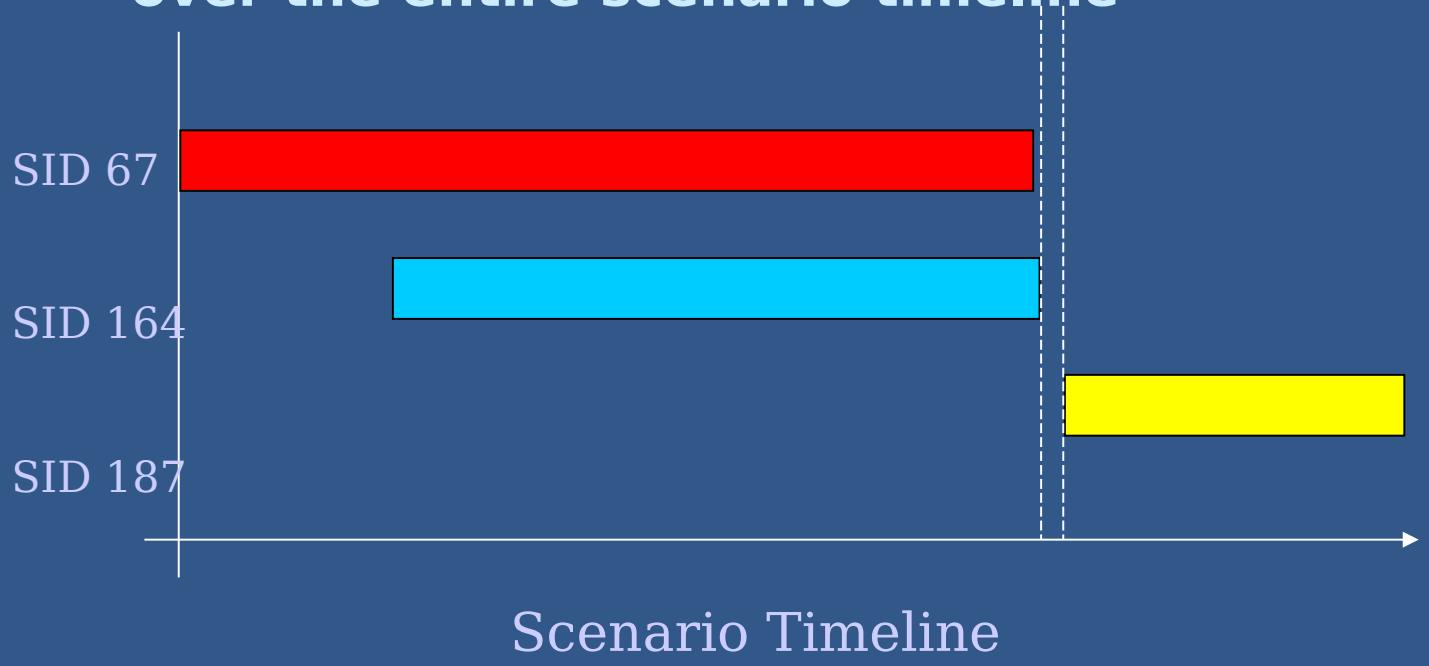
Radar Coverage Reports

- **S/N Plots:**
  - **Using sensor model, sensor entity, vehicle waypoints, and vehicle RCS, a S/N graph is generated**



Radar Coverage Reports

- **Vehicle Coverage Reports:**
  - **User selects VID for radar coverage report**
  - **Using S/N data for this VID, a coverage report is generated for the selected vehicle over the entire scenario timeline**



Vehicle Information Table

VID/SID	Description	Call Sign	Mode 2	Mode 3
23/148	SFIT	USS Anzio	0122	0133
90	SWRB	Weapons Recovery	0432	0239
100	F/A-18A	Fighter Aircraft	0272	0444
110	TARGET	Target		
120				

Timeline Enhancements